MS244D

C-887 Hexapod Controller

User manual

Version: 1.4.0 Date: 26.01.2023

This document describes the following 6-axis controllers for hexapods:

- **C-887.52**
- **C-887.521**
- **C-887.522**
- **C-887.523**
- **C-887.53**
- **C-887.531**
- **C-887.532**
- **C-887.533**

Physik Instrumente (PI) GmbH & Co. KG, Auf der Roemerstrasse 1, 76228 Karlsruhe, Germany
Phone +49 721 4846-0, Fax +49 721 4846-1019, Email info@pi.ws, www.pi.ws

The following company names and brands are registered trademarks of Physik Instrumente (PI) GmbH & Co. KG:

PI®, NanoCube®, PICMA®, PILine®, NEXLINE®, PiezoWalk®, NEXACT®, Picoactuator®, PInano®, PIMag®, Q-Motion®

Notes on brand names and third-party trademarks:
Microsoft® and Windows® are registered trademarks or trademarks of Microsoft Corporation in the USA and/or other countries.
EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
TwinCAT® is a registered trademark of and licensed by Beckhoff Automation GmbH.
LabVIEW, National Instruments and NI are trademarks of National Instruments. Neither the driver software nor the software programs offered by PI or other goods and services are connected to or sponsored by National Instruments.
Python® is a registered trademark of Python Software Foundation.
BiSS is a registered trademark of iC-Haus GmbH.
The following designations are protected company names, trademarks or registered trademarks of other owners:
Linux, MATLAB, MathWorks

The patents held by PI are found in our patent list: https://www.physikinstrumente.com/en/about-pi/patents

The software products provided by PI are subject to the General Software License Terms of Physik Instrumente (PI) GmbH & Co. KG and may contain and/or use third-party software components. Further information can be found in the General Software License Terms (https://www.physikinstrumente.com/download/EULA_PhysikInstrumenteGmbH_Co_KG.pdf) and in the Third-Party Software Notes (https://www.physikinstrumente.com/download/TPSWNote_PhysikInstrumenteGmbH_Co_KG.pdf) on our website.

Original instructions
First printing: 26.01.2023
Document number: MS244D, BRo, Version 1.4.0

Subject to change. This manual is superseded by any new release. The latest respective release is available for download on our website (https://www.pi.ws).

Contents

1 About this Document

In this Chapter

1.1 Objective and Target Audience of this User Manual

This user manual contains the information required for using the C-887 as intended.

It assumes that the reader has a fundamental understanding of basic servo systems as well as motion control concepts and applicable safety procedures.

The latest versions of the user manuals are available for download on our website (p. 7).

1.2 Symbols and Typographic Conventions

The following symbols and typographic conventions are used in this user manual:

CAUTION

Dangerous situation
Failure to comply could lead to minor injury.

➢ Precautionary measures to avoid the risk.

NOTICE

Dangerous situation
If this situation is not avoided, it will damage the equipment.

➢ Precautionary measures to avoid the risk.

INFORMATION	
Information for easier handling, tricks, tips, etc.	

Symbol/Label	Meaning
1. 2.	Action consisting of several steps with strict sequential order
➢	Action consisting of one or more steps without relevant sequential order
▪	Bullet points
p. 5	Cross-reference to page 5
RS-232	Label on the product indicating an operating element (example: RS-232 interface socket)
⚠	Warning sign on the product referring to detailed information in this manual.
Start > Settings	Menu path in the PC software (example: to open the menu, the ***Start*** and ***Settings*** menu items must be clicked in succession)
`POS?`	Command line or a command from PI's General Command Set (GCS) (example: Command to get the axis position).
Device S/N	Parameter name (example: Parameter where the serial number is stored)
5	Value that must be entered or selected via the PC software

1.3 Figures

For better understandability, the colors, proportions, and degree of detail in illustrations can deviate from the actual circumstances. Photographic illustrations may also differ and must not be seen as guaranteed properties.

1.4 Definition of Terms

Absolute-measuring position sensor	Sensor (encoder) for capturing changes of position or changes of angle. Signals from the absolute-measuring position sensor are used for axis position feedback. After the controller is switched on, absolute target positions can be commanded and reached immediately. Referencing is not necessary.
Axis	Also referred to as "logical axis". Logical axes reproduce the translations and rotations of the motion platform of the hexapod and the motion of the optionally usable positioners in the firmware of the C-887. Each direction of motion corresponds to a logical axis. Further information on translations and rotations can be found in the manual for the hexapod. All motion commands of the C-887 refer to logical axes.
Orientational coordinate system	The orientational coordinate system can be used to change the direction of the translational axes X and/or Y and/or Z permanently (e.g., when Z is to always point in the direction of the default X axis). In the default setting, the PI_Base orientational coordinate system is active.
Workspace	The entirety of all combinations of translations and rotations that the hexapod can approach from the current position is referred to as the workspace. The workspace can be limited by the following external factors: ▪ Installation space ▪ Dimensions and position of the load
Operating coordinate system	With the operating coordinate system, the position display, direction of motion, and the center of rotation for the motion platform of the hexapod is adapted to the application. It is also possible to use --> work-and-tool coordinate systems. In the default setting, the operating coordinate system ZERO is active.
Center of rotation	The center of rotation describes the intersection of the rotational axes U, V, and W. The center of rotation always moves together with the platform. Depending on the active --> operating coordinate system, the center of rotation can be moved from the origin of the coordinate system in the X and/or Y and/or Z direction with the `SPI` command. The center of rotation that can be moved using the `SPI` command is also referred to as "pivot point".
Dynamics profile	Comprises the target position, velocity, and acceleration of the axis calculated for each point in time of the motion. The calculated values are called "commanded values". The dynamics profile can be created by the profile generator of the C-887 or by the wave generators, or it can be externally created and transferred to the C-887 through the cyclic transfer of target positions.

Fast alignment routine
The C-887 supports routines for fast alignment of a transmitter or receiver. The objective of the routines is to align the transmitter and receiver so that the maximum intensity of the transmitted signal is measured on the receiver side. The fast alignment routines include all routines that are defined with the FDR and FDG commands and are started with the FRS command.

Firmware
Software that is installed on the controller.

Volatile memory
RAM module where the parameters are saved when the controller is switched on (working memory). The parameter values in the volatile memory determine the current behavior of the system.
The parameter values in the volatile memory are also referred to as "Active Values" in the PC software from PI.

GCS
PI General Command Set; command set for PI controllers. Piezo drivers and servo controllers can be operated together with minimal programming effort thanks to GCS.

Hexapod strut
Several hexapod struts need to be moved in order to move a logical axis of the motion platform.

The C-887 calculates the target positions for the individual struts from the target positions given for the translational and rotational axes. The velocities and accelerations of the struts are calculated so that all struts start and stop at the same time.

Hexapod system
The combination of hexapod, controller, cables, and power adapter(s) is referred to as "hexapod system" in this manual.

Incremental position sensor
Sensor (encoder) for detecting changes of position or changes of angle. Signals from the incremental position sensor are used for axis position feedback. After the controller is switched on, referencing must be done before absolute target positions can be commanded and reached.

Coordinate system
The position display, direction of motion, and center of rotation for the motion platform of the hexapod are determined by coordinate systems that are linked in a chain. The chain is basically structured as follows (starting point > end point): --> HEXAPOD coordinate system, --> leveling coordinate system, --> orientational coordinate system, --> operating coordinate system.

The coordinate systems are always right-handed systems.

The HEXAPOD coordinate system determines the fundamental properties of all other coordinate systems. HEXAPOD is based on the configuration file with the geometrical data of the hexapod. The dimensional drawing in the manual of the hexapod shows the respective position of the HEXAPOD coordinate system.

Using the controller, custom coordinate systems can be defined and used instead of the default coordinate systems.

Based on HEXAPOD, the orientational and leveling coordinate systems adapt fundamental properties of the active operating coordinate

	system, and in most applications their custom definition and activation is not necessary at all or only once.
Leveling coordinate system	The leveling coordinate system can be used to correct errors in the orientation of the hexapod permanently (e.g., installation errors). In the default setting, the leveling coordinate system PI_Levelling is active.
PC software	Software installed on the PC.
Nonvolatile memory	Memory module (read-only memory, e.g., EEPROM or flash memory) where the default values of the parameters are loaded to the volatile memory when the controller is started. The parameter values in the nonvolatile memory are also referred to as "startup values" in the PC software from PI.
Pivot point	Depending on the active --> operating coordinate system, the center of rotation can be moved from the origin of the coordinate system in the X and/or Y and/or Z direction with the `SPI` command. The center of rotation that can be moved using the `SPI` command is also referred to as "pivot point".
Work and tool coordinate systems	The work-and-tool concept can be used for work with user-defined coordinate systems. The work-and-tool concept uses a combination of two active --> operating coordinate systems ("work coordinate system" and "tool coordinate system"). The X, Y, and Z axes of the tool coordinate system are permanently connected to the motion platform of the hexapod; i.e., the tool coordinate system moves together with the platform. The X, Y, and Z axes of the work coordinate system are always spatially fixed (in relation to the hexapod); i.e., the work coordinate system does **not** move when the platform of the hexapod moves. The current position of the motion platform of the hexapod can be considered the position of the tool coordinate system in the work coordinate system. The center of rotation always lies at the origin of the tool coordinate system and it therefore moves just as the tool coordinate system with the platform.

1.5 Other Applicable Documents

The devices and software tools from PI mentioned in this documentation are described in separate manuals.

The latest versions of the user manuals are available for download on our website (p. 7).

Supplementary documentation for the C-887:

Description	Document
Coordinate Systems for Hexapod Microrobots	C887T0007 User Manual
Hexapod motion Position and Orientation in Space, Center of Rotation	C887T0021 Technical Note
PI Hexapod Simulation Tool Determining the workspace and the permissible load for the hexapod	A000T0068 Software Manual
Fast, multichannel alignment in photonics	E712T0016 User Manual
For C-887.53, .531, .532, .533 models only: EtherCAT Interface Description	C887T0011 User Manual

Documentation for the available PC software:

Description	Document
GCS array data format description	SM146E Software Manual
GCS driver library for use with NI LabVIEW software	SM158E Software Manual
PI GCS 2.0 DLL	SM151E Software Manual
PI MATLAB Driver GCS 2.0	SM155E Software Manual
PIPython	SM157E User Manual
PIMikroMove	SM148E Software Manual
PIHexapodEmulator	C887T0001 User Manual
PIStages3Editor	SM156E User Manual
PI Update Finder: Searching for and downloading updates	A000T0028 User Manual

Basic information on EtherCAT networks and the CiA402 drive profile:

Description	Document
Adjustable speed electrical power drive systems - Part 7-201: Generic interface and use of profiles for power drive systems (PDS) - Profile type 1 specification (IEC 61800-7-201:2015)	DIN EN 61800-7-201:2016

Description	Document
Adjustable speed electrical power drive systems - Part 7-301: Generic interface and use of profiles for power drive systems - Mapping of profile type 1 to network technologies	DIN EN 61800-7-301:2016
EtherCAT implementation directive for CiA402 drive profile: Directive for using IEC 61800-7-201 within EtherCAT-based servo drives	ETG.6010 D (R) V1.1.0

Documentation for an EtherCAT example from PI:

Description	Document
Implementing a C-887 PI Controller in TwinCAT 3.1 for Motion and Activation of new Coordinate Systems	A000T0075 User Manual

User manuals for hexapod microrobots, e. g.:

Model family	Document
H-206 Hexapod for 6D Alignment and Micromanipulation	MS203E
H-810 Miniature Hexapod Microrobot	MS198E
H-811 Miniature Hexapod Microrobot	MS235E
H-820 Hexapod Microrobot	MS207E
H-824 Compact Hexapod Microrobot	MS200E
H-825 Compact Hexapod Microrobot	MS250E
H-840 Hexapod Microrobot	MS201E
H-850 Hexapod Microrobot	MS202E

1.6 Downloading Manuals

INFORMATION

If a manual is missing or problems occur with downloading:

➢ Contact our customer service department (p. 343).

Downloading manuals

1. Open the website **www.pi.ws**.
2. Search the website for the product number (e.g., C-887).
3. Click the corresponding product to open the product detail page.
4. Click the ***Downloads*** tab.

   The manuals are shown under ***Documentation***. Software manuals are shown under ***General Software Documentation***.
5. Click the desired manual and fill out the inquiry form.

   The download link will then be sent to the email address entered.

2 Safety

In this Chapter

2.1 Intended Use

The C-887 is a laboratory device as defined by DIN EN 61010-1. It is intended for indoor use and use in an environment that is free of dirt, oil, and lubricants.

According to its design, the C-887 is intended for closed-loop operation of a hexapod microrobot from PI that is equipped with drives with integrated motor drivers. As of serial number 121017873 the C-887 also supports hexapod microrobots that use the BiSS protocol for data transmission.

The C-887 may not be used for purposes other than those stated in this user manual.

The C-887 may only be used in compliance with the technical specifications and instructions in this user manual. The operator is responsible for validating the process.

2.2 General Safety Instructions

The C-887 is built according to state-of-the-art technology and recognized safety standards. Improper use of the C-887 may result in personal injury and/or damage to the C-887.

- Use the C-887 for its intended purpose only, and only when it is in perfect condition.
- Read the user manual.
- Immediately eliminate any faults and malfunctions that are likely to affect safety.

The operator is responsible for installing and operating the C-887 correctly.

2.2.1 Organizational Measures

User manual

- Always keep this user manual together with the C-887. The latest versions of the user manuals are available for download on our website (p. 7).
- Add all information from the manufacturer such as supplements or technical notes to the user manual.

- If you give the C-887 to other users, include this user manual as well as all other relevant information provided by the manufacturer.
- Do the work only if the user manual is complete. Missing information due to an incomplete user manual can result in minor injury and damage to equipment.
- Install and operate the C-887 only after you have read and understood this user manual.

Personnel qualification

The C-887 may only be installed, started, operated, maintained, and cleaned by authorized and appropriately qualified personnel.

2.2.2 Safety Measures during Installation

- Install the C-887 near the power adapter so that the power plug can be quickly and easily disconnected from the mains.
- Use the power cord supplied to connect the C-887 to the mains.
- If the supplied power cord needs to be replaced, use a sufficiently sized power cord.

Impermissible mechanical load and collisions between the hexapod, the load to be moved, and the surroundings can damage the hexapod.

- Only hold the hexapod by the base plate.
- Before installing the load, determine the limit value for the load of the hexapod with a simulation program (p. 60).
- Before installing the load, determine the workspace of the hexapod with a simulation program (p. 60).
- Make sure that the installed load observes the limit value determined with the simulation program.
- Avoid high forces and torques on the motion platform during installation of the hexapod and the load.
- To avoid unintentional deactivation of the hexapod system and resulting position changes of the hexapod, make sure that the power supply is not interrupted.
- Make sure that no collisions between the hexapod, the load to be moved, and the surroundings are possible in the workspace of the hexapod.

2.2.3 Measures during Startup and Operation

Risk of minor injuries from crushing between the moving parts of the hexapod and a stationary part or obstacle.

- ➢ Keep your fingers away from areas where they could be caught by moving parts.

When communication between the C-887 and the PC has been established via TCP/IP, the PC software offers all controllers present in the same network for selection. After a C-887 has been selected for the connection, all commands are sent to this controller. If the wrong controller is selected, unexpected motion of the hexapod could be commanded and cause minor crush injuries to the operating and maintenance staff.

- ➢ If several C-887 are displayed in the PC software, make sure that you select the right C-887.

Damage can occur to the hexapod if the transport safeguard of the hexapod has not been removed and a motion is commanded.

- ➢ Remove the transport safeguard before you start up the hexapod system.

Collisions can damage the hexapod, the load to be moved, and the surroundings.

General measures for avoiding collisions:

- ➢ Make sure that no collisions are possible between the hexapod, the load to be moved, and the surroundings in the workspace of the hexapod.
- ➢ Do not place any objects in areas where they can be caught by moving parts.
- ➢ Stop the motion immediately if a controller malfunction occurs.
- ➢ Note that the hexapod moves unpredictably during a reference move. A collision check or prevention does **not** take place. Soft limits that have been set for the motion platform of the hexapod with the NLM and PLM commands are ignored during the reference move.

Depending on the source of the dynamics profile, the platform of the hexapod can move along an undefined path under certain conditions. As a result, collisions are possible between the hexapod, the load to be moved, and the surroundings.

When the dynamics profile is determined by the profile generator of the C-887 (default):

- ➢ Avoid sending new target positions while the hexapod (axes X, Y, Z, U, V, W) is still moving.
- ➢ If new target positions have to be sent while the hexapod is still moving (axes X, Y, Z, U, V, W): Only use motion commands to set target positions that maximally deviate from the current position by the value of the ***Path Control Step Size*** parameter (ID 0x19001504).

When the dynamics profile is determined by consecutive MOV commands:

- Only set target positions whose distance from each other is maximally as large as the value of the ***Path Control Step Size*** parameter (ID 0x19001504) with consecutive MOV commands.

When scanning procedures are carried out with the commands AAP, FIO, FLM, FLS, FSA, FSC, and FSM, the platform of the hexapod moves along an undefined path and can tilt if the values for distances or angles are too large. As a result, collisions between the hexapod, the load to be moved, and the surroundings are possible, and the scanning procedure can end with an unsatisfactory result. Measures for avoiding tilting:

- Select suitable values for paths and angles. For the hexapod models H-810, H-811, and H-206, 0.2 mm or 0.2 degrees should not be exceeded; for other hexapod models as well as in the case of changed settings for coordinate systems and the pivot point, the ideal values have to be determined experimentally.
- Set the velocity for the motion platform of the hexapod as low as possible (with the VLS command).
- Align the motion platform suitably before starting the scanning procedure.
- Use suitable holders for the inputs and/or outputs of the optical element to be aligned on the motion platform so that the motion during the scanning procedure only takes place over short distances or angles.

During a scanning procedure started with the FSA command, the scanning area can increase up to double the original area. As a result, collisions are possible between the hexapod, the load to be moved, and the surroundings.

- Make sure that the platform can safely move outside of the originally specified scanning area as well.

When the actual load of the hexapod motion platform exceeds the maximum holding force based on the self-locking of the actuators, switching off the servo mode for the motion platform axes of the hexapod can cause unintentional position changes of the hexapod. As a result, collisions are possible between the hexapod, the load to be moved, and the surroundings.

- Make sure that the actual load of the hexapod motion platform does not exceed the maximum holding force based on the self-locking of the actuators before you switch off the servo mode, reboot, or switch off the C-887.
- To avoid unintentional deactivation of the hexapod system and resulting position changes of the hexapod make sure that the power supply is not interrupted.

Unsuitable parameter settings can lead to improper operation or damage to the connected mechanical system.

- Only change parameters after careful consideration.

2.2.4 Safety Measures during Maintenance

The C-887 contains electrostatic-sensitive devices (ESD) that can be damaged in the case of short circuits or flashovers.

- Before cleaning the housing, disconnect the C-887 from the power supply by removing the power plug.

Risk of minor injuries from crushing between the moving parts of the hexapod and a stationary part or obstacle.

- Keep your fingers away from areas where they could be caught by moving parts.

Collisions can damage the hexapod, the load to be moved, and the surroundings.

During a strut test, the hexapod moves unpredictably. A collision check or prevention does **not** take place, even if a configuration for preventing collisions was stored on the C-887 with the PIVeriMove hexapod software for collision checking. Soft limits that have been set for the motion platform of the hexapod with the `NLM` (p. 234) and `PLM` (p. 236) commands are ignored during a strut test. As a result, collisions are possible between the hexapod, the load to be moved, and the surroundings.

- Make sure that no collisions between the hexapod, the load to be moved, and the surroundings are possible during a strut test of the hexapod.
- Do not place any objects in areas where they can be caught by moving parts during a strut test.
- Observe the hexapod during a strut test in order to be able to intervene quickly in the case of malfunctions.

During a strut test, the hexapod strut can reach a limit switch. As a result, the servo mode is automatically switched off for the axes of the hexapod motion platform. Switching off the servo mode can cause unintentional position changes of the hexapod. As a result, collisions are possible between the hexapod, the load to be moved, and the surroundings.

- Make sure that the actual load of the motion platform of the hexapod does not exceed the maximum holding force based on the self-locking of the actuators before you start a strut test.

3 Product Description

In this Chapter

3.1 Model Overview

The C-887 hexapod controller is available in the following versions:

Model	Designation
C-887.52	6-axis controller for hexapods, TCP/IP, RS-232, benchtop device, incl. control of two additional axes
C-887.521	6-axis controller for hexapods, TCP/IP, RS-232, benchtop device, incl. control of two additional axes, analog inputs
C-887.522	6-axis controller for hexapods, TCP/IP, RS-232, benchtop device, incl. control of two additional axes, motion stop
C-887.523	6-axis controller for hexapods, TCP/IP, RS-232, benchtop device, incl. control of two additional axes, motion stop, analog inputs
C-887.53	6-axis controller for hexapods, TCP/IP, RS-232, benchtop device, incl. control of two additional axes, EtherCAT interface
C-887.531	6-axis controller for hexapods, TCP/IP, RS-232, benchtop device, incl. control of two additional axes, EtherCAT interface, analog inputs
C-887.532	6-axis controller for hexapods, TCP/IP, RS-232, benchtop device, incl. control of two additional axes, EtherCAT interface, motion stop
C-887.533	6-axis controller for hexapods, TCP/IP, RS-232, benchtop device, incl. control of two additional axes, EtherCAT interface, motion stop, analog inputs

INFORMATION

As of serial number 121017873 the C-887 also supports hexapod microrobots that use the BiSS protocol for data transmission.

3.2 Product View

3.2.1 Front Panel

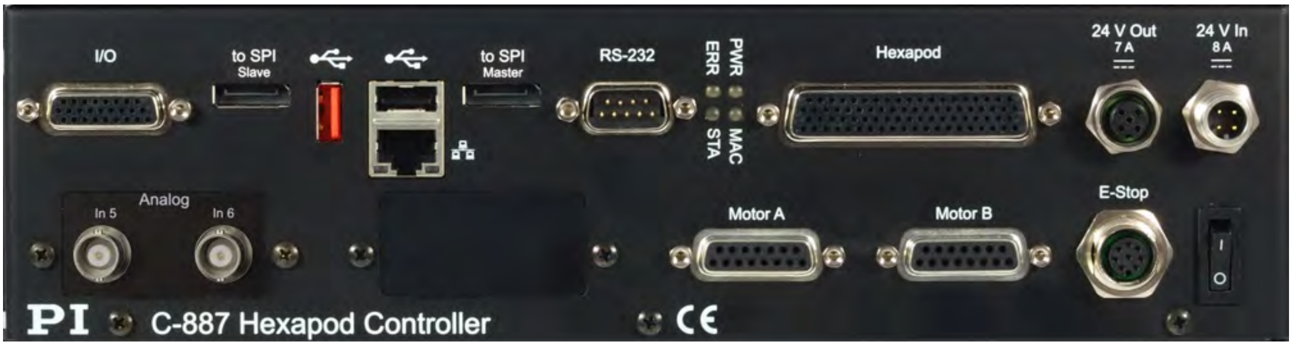
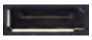
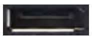
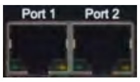


Figure 1: C-887.523 front panel; see table for availability of operating elements on other models

Figure 2: C-887.533 front panel; see table for availability of operating elements on other models

Labeling	Type	Function
I/O	HD D-sub 26 (f) (p. 354) 	Digital inputs/outputs: ▪ Outputs: Trigger external devices ▪ Inputs: Use in macros Analog inputs (multifunctional)

Labeling	Type	Function
to SPI Slave	DisplayPort 	Connector for an SPI slave. For PI internal purposes only.
	USB type A, for high insertion and pulling forces 	USB interface for connecting the C-887.MC2 or C-887.MC manual control units
	USB type A 	USB interface for connecting peripheral devices
	RJ45 socket 	Network connection via TCP/IP
to SPI Master	DisplayPort 	Connector for an SPI master. For PI internal purposes only.
RS-232	D-sub 9 (m) (p. 358) 	Serial connection to PC
ERR	LED red/off 	Error indicator: ▪ Lights up continuously: Error (error code ≠ 0) ▪ Off: No error (error code = 0) The error code can be queried with the ERR? command. The query resets the error code to zero and the LED is switched off.
PWR	LED green/off 	Power: ▪ Lights up continuously: Booting the firmware is complete and the controller is ready for normal operation. ▪ Off: The controller is switched off or the firmware is booting.
STA	LED green/off 	State: ▪ Lights up continuously: Booting the firmware is complete and the controller is ready for normal operation. ▪ Off: The controller is switched off or the firmware is booting.

Labeling	Type	Function
MAC	LED green/red/off 	Macro: ▪ Lights up green: Macro is running ▪ Lights up red: Macro error The error code can be queried with the `MAC ERR?` command. The query resets the error code to zero and the LED is switched off. ▪ Off: No macro is running and there is no macro error.
Hexapod	HD D-sub 78 (f) (p. 356) 	Connector for data transmission between hexapod and controller As of serial number 121017873 the C-887 also supports data transmission via the BiSS protocol.
24 V Out 7 A	4-pole M12 socket (f) (p. 354) 	Power supply for hexapod C-887.522, .523, .532, .533: The 24 V output for the hexapod must be activated via the **E-Stop** socket.
24 V In 8 A	4-pin M12 panel plug (m) (p. 353) 	Connector for the supply voltage of the C-887 When the hexapod is connected to the **24 V Out 7 A** socket of the controller, the supply voltage of the controller is also used for the hexapod.
Analog **In 5** **In 6**	C-887.521, C-887.523, .531, .533: BNC sockets  C-887.52, C-887.522, .53, .532: Covered with blank panel	C-887.521, .523, .531, .533 only: Analog inputs, -5 V to 5 V Refer to "Analog Input Specifications" (p. 348).
EtherCAT **Port 1** **Port 2**	C-887.53x**: RJ45 socket with green and yellow LED  C-887.52x*: Covered with blank panel	C-887.53x** only: Port 1 (left): Connector for EtherCAT master Port 2 (right): Connector for the next EtherCAT slave Green LED: ▪ Lights up continuously: EtherCAT connection was established ▪ Flickers: EtherCAT slave is transmitting/receiving Ethernet frames ▪ Off: No EtherCAT connection Yellow LED: Not used

Labeling	Type	Function
RUN	C-887.53x**: LED green/off C-887.52x*: Covered with blank panel	C-887.53x** only: Communication status of the EtherCAT slave: ▪ Off: Slave is in the INIT state. ▪ Flashes (2.5 Hz): Slave is in the PRE-OPERATIONAL state (before operation) ▪ Single flash: Slave is in the SAFE OPERATIONAL state (in safe operation) ▪ Lights up continuously: Slave is in the OPERATIONAL state (in operation)
ERR	C-887.53x**: LED red/off C-887.52x*: Covered with blank panel	C-887.53x** only: Communication status of the EtherCAT slave: ▪ Off: No error, slave communicating via EtherCAT ▪ Flashes (2.5 Hz): Invalid configuration. General configuration error. Possible cause: A status change specified by the master is not possible due to register or object settings. ▪ Single flash: Local error. The slave application has independently changed the EtherCAT state. Possible cause 1: A host watchdog timeout has occurred. Possible cause 2: Synchronization error, the slave changes automatically to SAFE-OPERATIONAL. ▪ Double flash: A process data watchdog timeout has occurred. Possible cause: Sync manager watchdog timeout.
Motor A **Motor B**	D-sub 15 (f) (p. 357) 	Two connectors for positioners. Only for positioners with DC motor and integrated motor driver! ▪ PWM signal output for the positioner ▪ Input of the signals of the position sensor ▪ Input of the limit switch and reference switch signals
E-Stop	C-887.522, .523, .532, .533: 8-pole M12 socket (f) (p. 354)  C-887.52, .521, .53, .531: Covered with dummy plug	C-887.522, .523, .532, .533 only: Connector for external hardware (e.g., pushbuttons or switches): The connection controls an internal relay with N/O contact that deactivates or activates the 24 V output for the hexapod (**24 V Out 7 A** socket). If no external hardware is used, the C887B0038 shorting plug (included in the scope of delivery (p. 21)) must be connected to activate the 24 V output.

Labeling	Type	Function
-	Toggle switch 	On/off switch: ▪ O position: The controller is switched off ▪ | position: The controller is switched on If the hexapod is connected to the **24 V Out 7 A** socket of the controller, the hexapod is also switched on/off. C-887.522, .523, .532, .533: The 24 V output for the hexapod must be activated via the **E-Stop** socket.

* C-887.52x stands for C-887.52, .521, .522, .523

** C-887.53x stands for C-887.53, .531, .532, .533

3.2.2 Type Plate

Labeling	Function
	Data matrix code (example; contains the serial number)
C-887.521	Product name (example), the characters following the period refer to the model
PI	Manufacturer's logo
115040741	Serial number (example), individual for each C-887 Meaning of each position (from the left): 1 = internal information, 2 and 3 = year of manufacture, 4 to 9 = consecutive number
Country of origin: Germany	Country of origin
⚠	Warning sign "Pay attention to the manual!"
	Old equipment disposal (p. 361)
CE	CE conformity mark
WWW.PI.WS	Manufacturer's address (website)

3.2.3 Ground Connection

Figure 3: C-887 controller, ground connection

Labeling	Type	Function
⏚	M4 threaded pin	Ground connection If potential equalization is required, the C-887 can be connected to the grounding system.

3.3 Scope of Delivery

Item ID	Components
C-887	Hexapod controller according to your order
C-815.563	Crossover network cable
C-815.553	Straight-through network cable
C-815.34	Null modem cable for connecting to the PC via RS-232
C-501.24180M12	Separate 24 V wide input range power supply (180 W/7.5 A) for use with line voltages from 100 to 240 V AC and voltage frequencies of 50 or 60 Hz, with 4-pole M12 connector (f)
3763	Power cord
C-990.CD1	PI software CD for digital electronics
MS247EK	Short instructions for hexapod systems

Item ID	Components
Only for the models C-887.522, C-887.523, C-887.532, C-887.533:	
C887B0038	Shortening plug for the controller's **E-Stop** socket 

3.4 Optional Accessories

3.4.1 Overview

Order number	Description
C-887.5xxx	Different hexapod cable sets (p. 24)
C-887.MC2	▪ Manual control unit for hexapods, USB connector with 3 m connection cable, rotary knobs for all Cartesian axes, buttons for motion stop and referencing, position display ▪ C887T0036 User Manual The controller requires at least firmware component FW version 2.7.1.1 (p. 147) to be compatible with the C-887.MC2 control unit. If necessary, update the controller's firmware (p. 319). The previous model C-887.MC is no longer available, however, the controller will still support it.
C-887.VM1	▪ PIVeriMove hexapod software for collision checking ▪ C887T0002 User Manual

Order number	Description
C-887.MSB	Motion Stop Button For C-887 models with **E-Stop** socket  For details, see "Using the E-Stop Socket" (p. 91).
F-712.PM1	Optical power meter, 400-1550 nm wavelength range, to 1 mA input current, 20 kHz signal bandwidth, logarithmic output ±5 V, benchtop device, including power adapter For C-887 models with **Analog In 5** and **In 6** BNC sockets Refer to the manual for the optical power meter (MP165E) for details.
F-712.IRP1	High-resolution optical power meter with logarithmic output signal, 600 to 1700 nm wavelength range, input power of 230 pW to 1.3 mW, 6 kHz signal bandwidth, logarithmic output voltage of 0.1 to 1.6 V, single-channel benchtop device. For C-887 models with **Analog In 5** and **In 6** BNC sockets Refer to the manual for the optical power meter (MP192E) for details.
R-FMP-GSM	Gradient search mode. Firmware functionality to allow simultaneous execution of fast alignment routines for gradient search in several axes of motion. The gradient search is available by default for all hexapod systems with H-811 miniature hexapod robot. For details on gradient search, see the "Fast Multi-Channel Photonics Alignment (FMPA)" manual (E712T0016).
L-xxx **M-xxx**	PI positioner equipped with DC motor and PWM amplifier. Information on suitable models on request.

➢ To order, contact our customer service department (p. 343).

3.4.2 Hexapod Cables

For information on the suitability of the cables for specific hexapod models, refer to "Cable Specifications" (p. 352).

Data transmission cable for hexapods, drag chain compatible, HD D-sub 78 m/f	
Order number	**Data transmission cable, available lengths**
C-815.82D02	2 m
C-815.82D03	3 m
C-815.82D05	5 m
C-815.82D07	7.5 m
C-815.82D10	10 m
C-815.82D20	20 m

Power supply cable for hexapods, drag-chain compatible, M12 m/f angled	
Order Number	**Power Supply Cables, Available Lengths**
C-815.82P02A	2 m
C-815.82P03A	3 m
C-815.82P05A	5 m
C-815.82P07A	7.5 m
C-815.82P10A	10 m
C-815.82P20A	20 m

Power supply cable for hexapods, drag chain compatible, M12 m/f straight	
Order number	**Power supply cable, available lengths**
C-815.82P02E	2 m
C-815.82P03E	3 m
C-815.82P05E	5 m
C-815.82P07E	7.5 m
C-815.82P10E	10 m
C-815.82P20E	20 m

To order, contact our customer service department (p. 343).

3.4.3 C-887.VM1 PIVeriMove as an Option for Collision Checking

The PIVeriMove hexapod software for collision checking, optionally available under the order number C-887.VM1 (p. 22), offers the following functions:

- Simulation of the arrangement of the hexapod and its surroundings on a PC
- Transfer of the configuration created on the PC to the C-887

With the created configuration, it is checked whether collisions occur between the following groups for any desired target positions:

- Surroundings including base plate of the hexapod
- Hexapod struts
- Motion platform of the hexapod including load

INFORMATION

The C-887 only carries out collision checks based on a configuration created with PIVeriMove when the absolute position of the mechanics is known (hexapod with incremental sensors: After a successful reference move).

The PIVeriMove hexapod software for collision checking is installed on a PC and activated there via a license key. For further information on installation and use, refer to the C887T0002 user manual (included in the scope of delivery of PIVeriMove).

3.5 Commandable Elements

The following table contains the elements that can be commanded with GCS commands (p. 133).

INFORMATION

The C-887.53, .531, .532, and .533 models are equipped with an EtherCAT interface. When the hexapod system is commanded via the EtherCAT interface, it is used as a multi-axis device according to the CiA402 drive profile. The EtherCAT master commands the logical axes X, Y, Z, U, V, and W of the hexapod motion platform.
For further information, see "Commanding via the EtherCAT Interface" (p. 44).

Element	Number	Identi-fier	Description
Logical axis	6	X, Y, Z, U, V, W	The logical axes X to W reproduce the translations and rotations of the motion platform of the hexapod in the firmware of the C-887. Translational axes: X, Y, and Z Rotational axes: U, V, and W

<table>
<tr><th>Element</th><th>Number</th><th>Identi-fier</th><th>Description</th></tr>
<tr><td>Logical axis</td><td>2</td><td>A, B</td><td>The logical axes A and B reproduce the motion of additional positioners in the firmware of the C-887.</td></tr>
<tr><td>Hexapod strut</td><td>6</td><td>1 to 6</td><td>The hexapod struts are not accessible for motion commands but can be selected as data sources for recording with the data recorder for diagnostic purposes, see also <code>DRC</code> (p. 157) and <code>HDR?</code> (p. 189).</td></tr>
<tr><td>Input signal channel</td><td>Maximum of 6</td><td>1 to 6</td><td>Channels of the analog inputs; the identifier is assigned to the input signal channels in the following order:
1 to 4: Analog inputs on the I/O socket (p. 354)
5 and 6: For the C-887.521, .523, .531, .533 models only. Analog inputs on the BNC sockets Analog In 5 (identifier 5) and In 6 (identifier 6)
The <code>TAC?</code> command (p. 254) gets the number of installed analog inputs.</td></tr>
<tr><td>Digital output</td><td>4</td><td>1 to 4</td><td>1 to 4 identify digital output lines 1 to 4 of the I/O socket (p. 354).
The <code>DIO</code> command (p. 155) sets the status of the digital output lines.</td></tr>
<tr><td>Digital input</td><td>4</td><td>1 to 4</td><td>1 to 4 identify digital input lines 1 to 4 of the I/O socket (p. 354).
The <code>DIO?</code> command (p. 156) gets the status of the digital input lines.</td></tr>
<tr><td>Wave generator</td><td>8</td><td>1 to 8</td><td>Each wave generator (p. 99) is permanently allocated to a logical axis:
<table>
<tr><th>Mechanics</th><th colspan="6">Hexapod</th></tr>
<tr><td>Axis</td><td>X</td><td>Y</td><td>Z</td><td>U</td><td>V</td><td>W</td></tr>
<tr><td>Wave generator</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr>
</table>
<table>
<tr><th>Mechanics</th><th colspan="2">Additional axes</th></tr>
<tr><td>Axis</td><td>A</td><td>B</td></tr>
<tr><td>Wave generator</td><td>7</td><td>8</td></tr>
</table></td></tr>
<tr><td>Wave table</td><td>100</td><td>1 to 100</td><td>The wave tables contain the saved data (a total of 10.000.000 points) for the waveforms output by the wave generators.
The value of the Number Of Waves parameter (ID 0x1300010A) indicates the number of wave tables (p. 99).</td></tr>
</table>

Element	Number	Identi-fier	Description
Coordinate system	unlimited	Name is assigned when the coordi-nate system is defi-ned	Using the controller, custom coordinate systems can be defined and used instead of the default coordinate systems. For more information, refer to "Hexapod Motion" (p. 31). **Naming conventions for coordinate systems:** Permissible characters: 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ_ The number of characters is unlimited. The name must start with an alphabetic character. Reserved names which may not be used for defining, copying or deleting: HEXAPOD, PI_LEVELLING, PI_BASE, ZERO, 0, NULL, XML, KLF, KLF(USER), KLF(PI), KLD, KLD(USER), KLD(PI), KSB, KSB(USER), KSB(PI), KSD, KSF, KST, KSW Each name can exist only once. Any existing coordinate system not in use will be overwritten when a coordinate system with the same name is created (defining, generating a copy).
Fast alignment routines	100	The name is assigned when the rou-tine is defined	The fast alignment routines include all routines that are defined with the FDR and FDG commands. **Conventions for names of fast alignment routines:** String consisting of alphanumeric characters. Spaces and special characters are not permitted. You can find further information in the "Fast Multi-Channel Photonics Alignment (FMPA)" document (E712T0016).
Data recorder table	16	1 to 16	The data recorder tables contain the recorded data. The C-887 has 16 data recorder tables (query with TNR? (p. 257)) with max. 262144 data points per table (default: 8192 data points per table). The number of points per data recorder table can be set with the ***Data Recorder Points Per Table*** parameter (ID 0x16000201).
Overallsystem	1	1	C-887 as an overall system.

3.6 Important Components of the Firmware

The firmware of the C-887 provides the following functional units:

Firmware component	Description
ASCII commands	Communication with the C-887 can be managed using the commands of the PI General Command Set (GCS; version 2.0). The GCS is independent of the hardware (controller, positioners connected). Examples of the use of GCS: ▪ Starting hexapod motion ▪ Getting system and motion values ▪ Defining a coordinate system ▪ Defining fast alignment routines You can find a list of the available commands in the "Command Overview" section (p. 135).
Configuration files, positioner databases, parameter	The settings that are used to adapt the C-887, e.g., to the properties of the mechanics or the communication interfaces used are determined by: ▪ Configuration files: see "Updating Firmware and Configuration Files" (p. 318) ▪ Positioner databases: Refer to "Operating Parameters for Axes A and B" (p. 30) ▪ Interface parameters: See "Establishing Communication via the TCP/IP Interface" (p. 72) and "Establishing Communication via the RS-232 Interface" (p. 78) ▪ For further parameters, see "Adapting Settings" (p. 299)
Command levels	The availability of commands and the write permission for the parameters are determined by command levels. The current command level can be changed with the CCL command. This may require entering a password. Refer to "Adapting Settings" (p. 299) for further information.
Data recorder	The C-887 contains a real-time data recorder (p. 97). This can record different input and output signals (e.g. current position, commanded position) from different data sources (e.g. logical axes, input signal channels) (p. 97).
Wave generator	Each logical axis can be controlled by a wave generator that outputs waveforms. The wave generator is especially suited for dynamic applications in which periodic motions of the axis are executed, for example (p. 99).
Macros	The C-887 can save macros. Command sequences can be defined and permanently stored via the macro function. A startup macro can be defined that runs each time the C-887 is switched on or rebooted. This simplifies operation without a connected PC. Further information can be found in the "Controller Macros" section (p. 122).

The firmware can be updated with a tool (p. 318).

3.7 ID Chip Detection

The hexapod has an ID chip that contains data on the type of hexapod, its serial number, and the date of manufacture. The data is loaded from the ID chip when the controller is switched on or rebooted. Depending on the data loaded, the controller keeps the current configuration or installs a new configuration.

For simple replacement, the configuration data for all standard hexapods is stored at the factory in every standard controller (e.g., geometry data and control parameters). The configuration data for customized hexapods is only stored on the controller if the hexapod and controller are delivered together, or if PI was correspondingly informed before delivery of the controller.

Depending on the data that is loaded from the ID chip, the controller behaves as follows after it is switched on or rebooted:

- If the hexapod type and the serial number loaded from the ID chip are identical with the data saved in the controller, the controller keeps the current configuration. The system is immediately ready for operation.
- If the hexapod type and/or the serial number loaded from the ID chip differ from the data saved in the controller:
  - If the hexapod type is identical but the serial number differs, the controller installs the standard configuration for this type.
  - If the hexapod type differs, the controller installs a new configuration that matches the new hexapod type.

  It can be necessary to reboot the controller to activate the installed configuration.

INFORMATION

Application notes for the ID chip detection:

- Before you replace the connected hexapod, save the current parameter values of the controller on the PC (p. 301).
- Connect the hexapod only when the controller is switched off.
- When the firmware has finished booting, send the `CST?` command (p. 154) to check whether the installed configuration has to be activated by rebooting the controller. A reboot is necessary when the response is "NOSTAGE". The controller can be rebooted with the `RBT` command (p. 239).
- Send the `ERR?` command (p. 164) to check whether the configuration was activated successfully. If the response to ERR? contains the error code 233 or 211, the configuration for the new hexapod is not in the controller (possible for example, for customized hexapods or new standard hexapods). Contact our customer service department (p. 343) in order to receive a suitable configuration file. For the installation of the new configuration file, see "Updating Firmware and Configuration Files" (p. 318).
- Send the `VER?` command (p. 261) to check the information for the hexapod type, serial number, and manufacturing date saved on the ID chip. Example for the response:
  `IDChip: H-811.F-2 SN123456789 20/1/2016`

3.8 Operating Parameters for Axes A and B

The **Motor A** and **Motor B** sockets (D-sub 15 (f)) of the C-887 are intended for connecting positioners with DC motor and integrated motor drivers.

If you inform PI on the positioner types used before the hexapod system is delivered, PI will configure the C-887 according to your order, so that the corresponding positioner types are assigned to axes A and B of the C-887:

- If you order only one positioner, the corresponding type is assigned to axis A.
- If you order two positioners, the corresponding types are assigned to axes A and B in ascending alphabetical order, for example, M-403.1DG to A and M-403.2PD to B; M-414.1PD to A and M-511.DD1 to B.

The axes A and B are deactivated and will not be visible in the PC software (e.g. in PIMikroMove) if their positioner type is set to NOSTAGE.

To change the positioner type's assignment, either use PI's PC software (e.g., PIMikroMove, see "Starting Motion" (p. 81)) or send the CST command (p. 153).

When assigning a new positioner type, its operating parameters will be loaded from one of the two following positioner databases:

File name	PISTAGES3.DB	PIStages2.dat
Memory location	PC	C-887
Description	Delivery includes parameter sets for standard positioners from PI and PI miCos; is automatically saved to the PC when the PC software is installed. New parameter sets can be created, edited, and saved.	Includes parameter sets for standard positioners from PI and PI miCos. Contact our customer service department (p. 343) if you want to update the PIStages2.dat in the C-887.
Necessary conditions for the use of the database	All conditions must be fulfilled: ▪ The C-887 supports the 0x3C parameter (from firmware version 2.5.2.1). ▪ Version 3.17.0 or newer of the dynamic program library for GCS (PI GCS 2.0 DLL) is available on the PC. ▪ The PI_qVST() and PI_CST() functions of the PI GCS 2.0 DLL are used. Example: PIMikroMove calls these functions if you assign a positioner type in the ***Start up controller*** window or send the CST and VST? commands in the ***Command entry*** window.	At least one condition must be fulfilled: ▪ The C-887's firmware version is older than 2.5.2.1. ▪ The version of the dynamic program library for GCS (PI GCS 2.0 DLL) is older than 3.17.0. ▪ You send the CST and VST? commands in a terminal program without using the PI GCS 2.0 DLL.

It is recommended to update the firmware and PC software versions in order to use the PISTAGES3.DB database. The PIStages2.dat is only made available for compatibility reasons.

The versions used in your hexapod system can be found in the response to the VER? command (p. 261).

Further information on positioner databases can be found in the manuals for PIStages3Editor and the GCS 2.0 DLL program library.

3.9 Motion of the Hexapod

3.9.1 Introduction

The C-887 hexapod controller is used to control a hexapod in six degrees of freedom with a very high positioning accuracy. A hexapod offers linear motion in the direction of the X, Y, and Z axes as well as rotations around each of these three axes.

The C-887 drives the motors of the six hexapod struts in closed-loop operation. The hexapod struts carry the motion platform and bring it into the desired position.

Positioning commands use Cartesian coordinates. The C-887 converts these into the respective positions and velocities of the hexapod struts before the platform moves to the desired position.

3.9.2 Travel Range and Soft Limits

Use TRA? (p. 257) to query the maximum absolute position that can be commanded when the platform of the hexapod moves along a specified direction vector. The maximum commandable position is calculated from the current position and it can be queried only if the platform of the hexapod is not in motion. Included in the calculation are the current settings for the soft limits (see NLM (p. 234), PLM (p. 236), SSL (p. 248)) and for the pivot point defined with SPI (see SPI (p. 245)) if it is used by the active operating coordinate system.

INFORMATION

Unavoidable rounding errors in the internal position calculation with TRA?

If the direction vector specified with TRA? would move the platform close to one of the travel range limits, due to rounding the response may indicate a position that cannot be reached. Commanding such a position fails and generates error code 7 ("Position out of limits"). Therefore, you can limit the response to TRA? with a factor so that only positions that can actually be commanded are displayed:

- Use SPA to set the ***Reduction Factor for TRA? Response*** parameter (0x19006000) to a suitable value between 0 and 1.

If permitted for the coordinate system currently being used, you can query the minimum and maximum commandable position of the individual axes with TMN? (p. 255) and TMX? (p. 256). Note that the travel ranges in X, Y, Z, U, V, W are dependent on each other. Depending on the current position of the motion platform of the hexapod, the actually available travel range for axes X, Y, Z, U, V and W can be smaller than the travel range displayed in the responses to TMN? and TMX?. The responses to TMN? and TMX? only correspond to the actually available travel range of an axis when the following conditions are met:

- All other axes are at zero position.
- The factory-set coordinate systems are active.
- The default settings for the pivot point coordinates apply.

Use VMO? (p. 263) to query whether a target position can be reached.

The physical unit of the position can be queried with PUN? (p. 238).

3.9.3 Supported Motion Types

The C-887 supports the following motion types:

Motion type	Triggering motion
Reference move (p. 174)	FRF Manual control unit (C887.MC2 or C-887.MC) (p. 22) for the axes of the hexapod motion platform (X, Y, Z, U, V, W)
	C-887 models with EtherCAT interface only: Homing mode according to CiA402 drive profile. The ***Configure Command Mode*** parameter (ID 0x19002000) requires the value 1 = "External: EtherCAT". Refer to "Commanding via the EtherCAT Interface" (p. 44) for further information.
Point-to-point motion; profile generator creates the dynamics profile (p. 34)	MOV: Motion to absolute target position The ***Trajectory Source*** parameter (ID 0x19001900) must have the value 0 (default).
	STE, IMP: Start jump or impulse, with data recording MVR: Motion relative to the last commanded target position MRT, MRW: Moves the specified axis relatively in the tool or work coordinate system. Refer to "Definition of Terms" (p. 3) for more details on coordinate systems
	Manual control unit (C-887.MC2 or C-887.MC) for the axes of the hexapod motion platform (X, Y, Z, U, V, W The rotary knobs of the control unit start motions. The knobs can be turned in increments. A rotation by one step starts motion by the step size that was set with the SST command.
Cyclic transfer of target positions (p. 36)	Consecutive MOV commands The ***Trajectory Source*** parameter (ID 0x19001900) must have the value 1 = "Dynamics profile is determined by consecutive MOV commands".
	C-887 models with EtherCAT interface only: Cyclic synchronous position (CSP) mode according to the CiA402 drive profile.

Motion type	Triggering motion
	The ***Configure Command Mode*** parameter (ID 0x19002000) requires the value 1 = "External: EtherCAT". Refer to "Commanding via the EtherCAT Interface" (p. 44) for further information.
Wave generator (p. 99)	WGO: Starts/stops the wave generator output
Fast alignment routines (p. 3)	You can find further information in the "Fast Multi-Channel Photonics Alignment (FMPA)" document (E712T0016).
Scanning procedures	AAP, FIO, FLM, FLS, FSA, FSC, FSM commands

INFORMATION

The C-887 can save and process command sequences as controller macros (p. 122).

All commands can be sent via the communication interfaces of the C-887 when a macro is running on the C-887. The macro content and commands received via the communication interfaces can overwrite each other.

INFORMATION

For axes with incremental sensors, motion can only be commanded after a successful reference move (p. 174) (also referred to as "initialization").

The behavior of the axes of the hexapod after the reference move is determined by the ***Behaviour After Reference Move*** (ID 0x07030401) and ***Target For Motion After Reference Move*** parameters (ID 0x07030402). Depending on the parameter values, the axes of the platform can be moved automatically e.g., to a specified position after the reference move.

- Value of the parameter 0x07030401 = 0: The axis remains in the reference position after the reference move.
- Value of the parameter 0x07030401 = 1: After the reference move, the axis moves to the absolute target position, which is specified by parameter 0x07030402.

A reference move is not required for axes with absolute-measuring sensors. The use of the FRF command is still recommended for these axes, however. FRF does **not** start a reference move for axes with absolute-measuring sensors but sets the target positions to the current position values. The above-described parameter values also go into effect, so that the axes can be moved to a defined "initial position", for example.

INFORMATION

If a manual control unit (C-887.MC2 or C-887.MC) (p. 22) is being used: The value of the ***HID Device Button Mode*** parameter (ID 0x0E001600) specifies the behavior of the "stop" and "reference move" pushbuttons:

- The parameter value is 0 (default): The pushbuttons trigger the respective action.
- The parameter value is 1: The pushbuttons do not trigger any actions. This can be a good idea for example, when the status of the pushbuttons is to be evaluated in a macro (query with the HIB? command (p. 191)).

The value of the parameter can be changed with the SPA command (p. 244) and saved with the WPA command (p. 276); see "Adapting Settings" (p. 299).

Refer to the documentation for the manual control unit for further information.

3.9.4 Profile Generator for Point-to-Point Motion

For point-to-point motion, the profile generator of the C-887 determines the dynamics profile.

INFORMATION

During a point-to-point motion, any new motion command sets the target position to a new value and the motion platform immediately approaches the new target position on an undefined path.

Parameters for the profile generator

Parameter	Description and Possible Values
Path Control Step Size (mm) 0x19001504	Step size for calculating the dynamics profile of the platform This parameter is write-protected and is adapted to the hexapod of the system before delivery.
Trajectory Velocity (Phys. Unit/s) 0x19001510	Velocity for the motion platform of the hexapod Changing the velocity with the VLS command overwrites the value of the parameter in the volatile memory.
Trajectory Acceleration (Phys. Unit/s2) 0x19001511	Acceleration for the motion platform of the hexapod
Trajectory Jerk (Phys. Unit/s3) 0x19001512	Jerk for the motion platform of the hexapod
Trajectory Source 0x19001900	Source of the dynamics profile for MOV commands For point-to-point motion triggered by the MOV command, the parameter must have the value 0 (default).

Commands for the profile generator

Command	Syntax	Function
`VLS`	VLS <SystemVelocity>	Sets the velocity for the motion platform of the hexapod. Limited by the ***Maximum System Velocity (Phys. Unit/s)*** parameter (ID 0x19001500) and ***Minimum System Velocity (Phys. Unit/s)*** parameter (ID 0x19001501).

Check of the dynamics profile created by the profile generator

When the dynamics profile for the hexapod is created by the profile generator, it is checked before the start of each motion whether the motion platform can actually reach the nodes of the calculated profile and the commanded target position. If a node or the target position cannot be reached, the motion is not executed. The following is checked:

- Are the nodes and the target position outside of the travel range limits that can be queried by `TMN?` (p. 255) and `TMX?` (p. 256) or `TRA?` (p. 257)?
- Have the soft limits set with `NLM` (p. 234) and `PLM` (p. 236) been activated with `SSL` (p. 248), and if yes, are the nodes and the target position outside of these soft limits?
- Are the individual drives able to move the platform to the required nodes and the specified target position?
- When a configuration for collision avoidance has been stored in the C-887 with the optionally available PIVeriMove hexapod software for collision checking: Do collisions occur between the following groups?
  - Surroundings incl. base plate of the hexapod
  - Drives of the hexapod
  - Motion platform of the hexapod incl. load

The `VMO?` command (p. 263) queries whether a specified target position can be reached.

INFORMATION

If motion commands generate error code 7 ("position out of limits") although even though the commanded target position is permissible, the motion platform may be at a position outside the travel range limits.

➢ If the hexapod is equipped with absolute-measuring sensors, temporarily deactivate the automatic check of the dynamics profile in order to be able to command the motion platform back to a permissible position. For details, refer to "Troubleshooting" (p. 335).

3.9.5 Cyclic Transfer of Target Positions

Options for specifying the dynamics profile through the cyclic transfer of target positions:

- Specification through consecutive MOV commands

  For details, see the following descriptions of parameters and commands

- Only C-887 models with EtherCAT interface: Cyclic synchronous position (CSP) mode according to the CiA402 drive profile.

  For further information, see "Commanding via the EtherCAT Interface" (p. 44).

NOTICE

Cyclic transfer of target positions!
Acceleration/deceleration, velocity, and steadiness of the motion depend on the following factors during the cyclic transfer of target positions:

- Target position values
- Observance of the cycle time

The execution of an unsuitable dynamics profile can tilt the hexapod. Tilting can damage the hexapod and/or the load affixed to it.

➢ For this reason, observe the following during the cyclic transfer of target positions:
  – The path that is specified by the target positions must be continuously differentiable at least twice.
  – During the execution of the dynamics profile, the maximum permissible velocity and acceleration of the hexapod must **not** be exceeded.
  – To generate the target positions and continuously transfer them to the C-887 during the motion, it is recommended to use a suitable program.

➢ If you specify the target positions by consecutive MOV commands: Use the buffer of the C-887 to make sure that the cycle time is adhered to:
  – Store the dynamics profile in the buffer of the C-887 before execution. For this purpose, use SPA to set the ***Trajectory Execution*** parameter (ID 0x19001901) to the value 1.
  – For a sensible use of the buffer, increase the value of the ***Threshold for Trajectory Execution*** parameter (0x19001903) with SPA (default = 1).

INFORMATION

Recommendation: When the dynamics profile to be run is known, you should use the wave generator (p. 99) as an alternative to successive MOV commands.

Parameters for specifying the dynamics profile through consecutive MOV commands

Parameter	Description and Possible Values
Path Control Step Size (mm) 0x19001504	Step size for calculating the dynamics profile of the platform This parameter is write-protected and is adapted to the hexapod of the system before delivery. The distance between the target positions set with consecutive `MOV` commands may only be maximally as large as the value of the parameter 0x19001504, in order to avoid tilting the hexapod.
Trajectory Source 0x19001900	Source of the dynamics profile for MOV commands For the cyclic transfer of target positions by consecutive MOV commands, the parameter must have the value 1.
Trajectory Execution 0x19001901	Execution of the dynamics profile Determines how the dynamics profile defined by consecutive `MOV` commands is executed: 0 = Dynamics profile is executed immediately (default) 1 = Dynamics profile is stored in a buffer before execution This parameter is only evaluated when the parameter 0x19001900 has the value 1.
Maximum Number of Trajectory Points 0x19001902	Maximum number of dynamics profile points Indicates the maximum size of the buffer. This parameter is write-protected and only evaluated when the parameters 0x19001900 and 0x19001901 both have the value 1.
Threshold for Trajectory Execution 0x19001903	Threshold value for executing the dynamics profile Determines how many dynamics profile points have to be stored in the buffer (by consecutive `MOV` commands) until the execution of the dynamics profile begins. This parameter is only evaluated when the parameters 0x19001900 and 0x19001901 both have the value 1.
Current Number Threshold for Trajectory Execution 0x19001904	Shows the current number of dynamics profile points in the buffer. The parameter value is always 0 when the dynamics profile is determined by the profile generator or the dynamics profile defined by consecutive `MOV` commands is immediately executed. This parameter is write-protected.

Commands for specifying the dynamics profile through consecutive MOV commands

Command	Syntax	Function
MOV	MOV {<AxisID> <Position>}	Consecutive MOV commands specify the single target positions. Other motion commands and starting the wave generator output are not allowed.
SCT	SCT "T" <CycleTime>	Determines the cycle time for running a dynamics profile. The cycle time is used to calculate the velocity during motion so that the specified points of the dynamics profile are each reached exactly at the end of the time interval (where possible in compliance with the velocity and acceleration limits).
VLS	VLS <SystemVelocity>	Limits the velocities of the individual drives.
#11	#11	Queries the free memory space of the buffer, the contents of which determine the dynamics profile of the hexapod, when the parameters 0x19001900 and 0x19001901 both have the value 1.

3.9.6 Coordinate Systems

The position display, direction of motion, and center of rotation for the motion platform of the hexapod are determined by coordinate systems that are linked in a chain. The chain is basically structured as follows (starting point > end point): --> HEXAPOD coordinate system, --> leveling coordinate system, --> orientational coordinate system, --> operating coordinate system.

The coordinate systems are always right-handed systems.

The HEXAPOD coordinate system determines the fundamental properties of all other coordinate systems. HEXAPOD is based on the configuration file with the geometrical data of the hexapod. The dimensional drawing in the manual of the hexapod shows the respective position of the HEXAPOD coordinate system.

Using the controller, custom coordinate systems can be defined and used instead of the default coordinate systems.

Based on HEXAPOD, the orientational and leveling coordinate systems adapt fundamental properties of the active operating coordinate system, and in most applications their custom definition and activation is not necessary at all or only once.

With the operating coordinate system, the position display, direction of motion, and the center of rotation for the motion platform of the hexapod is adapted to the application. It is also possible to use --> work-and-tool coordinate systems. In the default setting, the operating coordinate system ZERO is active.

The most important commands for working with user-defined coordinate systems:

- KSD (p. 216): Define operating coordinate system by entering the offset values for the axes X, Y, Z, U, V, and W
- KSF (p. 218): Define operating coordinate system at the current position of the motion platform of the hexapod ("home coordinate system")
- KLN (p. 209): Link coordinate systems with each other
- KEN (p. 200): Activate coordinate systems
- WPA SKS (p. 276): Save the currently valid settings for coordinate systems
- DPA SKS (p. 156): Reset settings for parameters and coordinate systems to default settings

For the C-887.53, .531 .532, and .533 models, the EtherCAT interface (p. 44) is activated by default after switching on or rebooting. The coordinate systems used (work and tool coordinate system) are defined and activated via the EtherCAT interface by PI-specific (SDO). The settings for coordinate systems cannot be changed by GCS commands when the EtherCAT interface is activated.

INFORMATION

Coordinate systems can be conveniently defined, linked, activated, and saved in PIMikroMove.

The ***Define Home Coordinate System (KSF)*** and ***Manage Coordinate Systems ...*** buttons are available in the ***Positioner Platform*** window for this. The hexapod and the active coordinate system are graphically displayed on the ***Positioner 3D View*** card.

If the ***Positioner Platform*** window (normally docked) and the ***Positioner 3-D View*** card are **not** displayed in the main window:

- ➢ Display the ***Positioner Platform*** window with the ***C-887 > Show Positioner platform settings*** menu item.
- ➢ Display the ***Positioner 3-D View*** card with the ***C-887 > Positioner 3-D View > Show*** menu item.

Further information on coordinate systems and the work-and-tool concept can be found in the document "Coordinate Systems for Hexapod Microrobots" (C887T0007) and in "Definition of Terms" (p. 3).

The PI Hexapod Simulation Tool is recommended for testing user-defined coordinate systems. You can find further information in the "PI Hexapod Simulation Tool - Determining the Workspace and the Permissible Load of the Hexapod" document (A000T0068).

For an example of the use of user-defined coordinate systems, see "Example for Defining an Operating Coordinate System with the KSD Command" (p. 41).

3.9.7 Rotations

Rotations are made around the center of rotation. a specified rotation in space is calculated from the individual rotations in the order U -> V- > W. This takes place irrespective of whether the values for U, V, and W were explicitly specified with the current command or result from the previous commands.

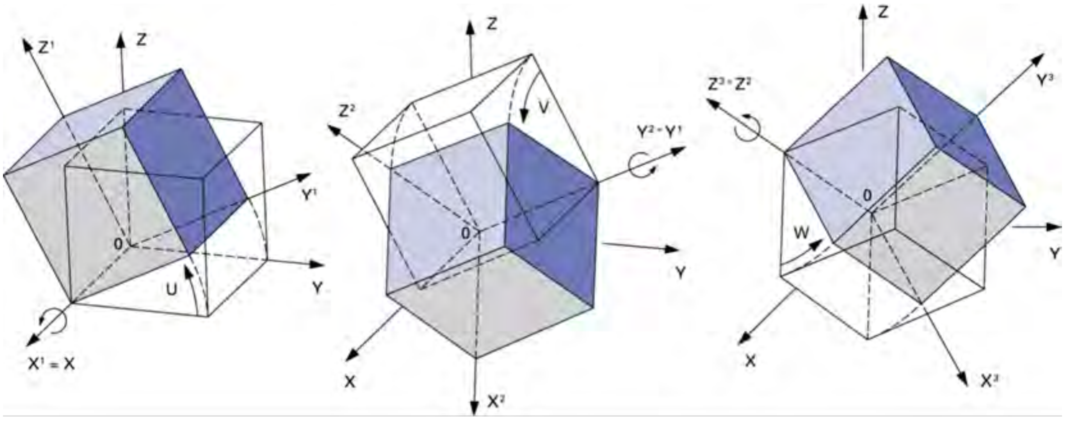


Figure 1: Order of elementary rotations of the hexapod platform when reaching a position (from left to right)

The center of rotation describes the intersection of the rotational axes U, V, and W.

The center of rotation always moves together with the platform.

The center of rotation can be changed at any time by defining and activating a KSD type operating coordinate system or work and tool coordinate systems (KSW and KST types).

The center of rotation can be moved from the origin of the tool coordinate system in the X and/or Y and/or Z direction with the SPI (p. 245) command when the active operating coordinate system is the ZERO coordinate system or a KSF type coordinate system and U = V = W = 0 applies to the rotational coordinates of the motion platform. The center of rotation that can be moved using the SPI command is also referred to as "pivot point".

For further information on calculating translations and rotations by the C-887, refer to the document "Hexapod Motion - Position and Orientation in Space, Center of Rotation" (C887T0021).

3.9.8 Example for Defining an Operating Coordinate System with the KSD Command

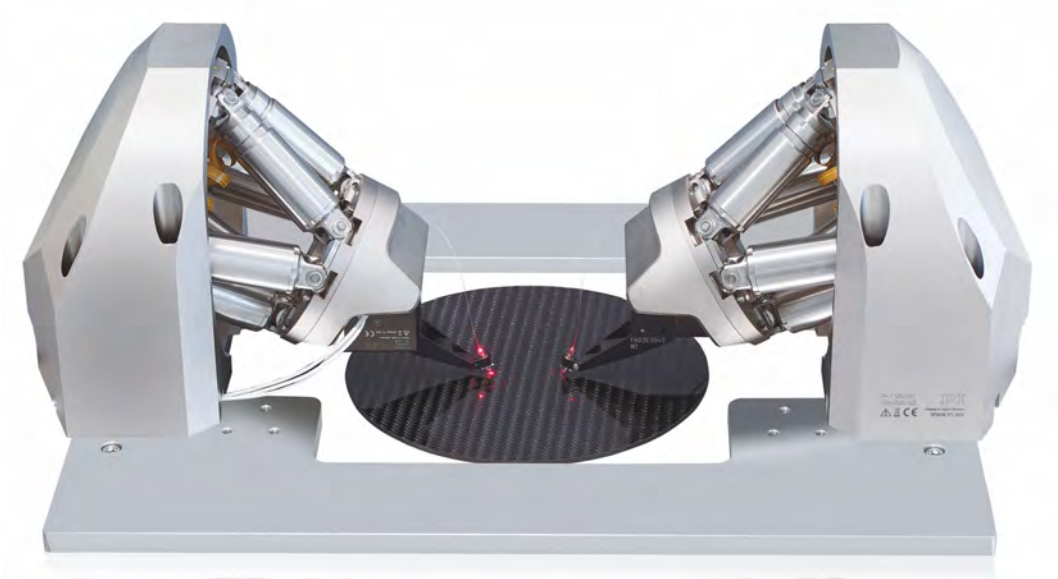


Figure 2: F-712.HA2 high-precision fiber alignment system: Application example

Customer-specific fiber holders are used in a high-precision F-712.HA2 fiber alignment system that contains one H-811 hexapod each on the transmitter and receiver side. The factory-set coordinate systems for transmitter and receiver are active upon delivery of the fiber alignment system (sender, receiver coordinate systems). However, the center of rotation for hexapod motion should be on the fiber tip each time. For this reason, customer-specific coordinate systems are defined in the hexapod controllers for the transmitter and receiver side. These systems are used to shift the center of rotation from the standard position to the fiber tip. In the following example, the distance between the default center of rotation and the fiber tip is 85.37 mm in the X direction and 71.88 mm in the Z direction; see also the figure below. The new coordinate systems are to be called fibertip1s (transmitter) and fibertip1r (receiver) and become immediately active when the fiber alignment system is switched on or rebooted.

The following commands are transmitted for the hexapod controller on the transmitter side:

Command	Function
KSD fibertip1s X 85.37 Z 71.88	Defines the new coordinate system
KLN fibertip1s sender	Links the new coordinate system to the sender coordinate system as child
KEN fibertip1s	Activates the new coordinate system
WPA SKS	Saves the settings in the nonvolatile memory

The default receiver coordinate system for the receiver side is rotated so that the direction of the X axis is inverted in relation to the transmitter side. For this reason, a negative offset value has to be specified for the X axis when the new coordinate system for the receiver side is defined. The following commands are transmitted for the hexapod controller on the receiver side:

Command	Function
KSD fibertip1r X -85.37 Z 71.88	Defines the new coordinate system
KLN fibertZp1r receiver	Links the new coordinate system to the receiver coordinate system as child
KEN fibertip1r	Activates the new coordinate system
WPA SKS	Saves the settings in the nonvolatile memory

The following figure shows the coordinate systems and their definition using the transmitter side as an example. For better understanding, not only is the resulting fibertip1s coordinate system displayed but also the other operating coordinate systems, from which fibertip1s takes its characteristics by linking: ZERO and sender (based on ZERO). The implementation of individual adapting steps as separate coordinate system definitions and the subsequent linking of these coordinate systems transfer the complex calculations for the resulting coordinate system into the C-887. The individual adapting steps can also be flexibly combined.

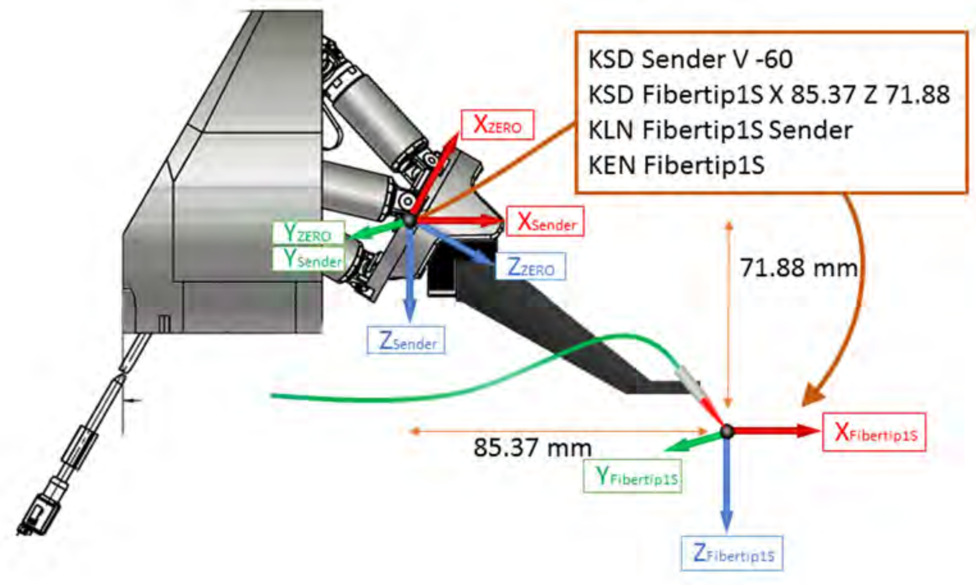


Figure 3: F-712.HA2: Shifting the center of rotation for the transmitter side to the fiber tip

3.9.9 Motion Status, Settling Window, Settling Time

Motion status

The C-887 shows the motion status and therefore attainment of the target position on the basis of status register bits. Status registers are available for the hexapod struts and axes A and B.

Register bits for displaying the motion status:

- Status register for each hexapod strut and axes A and B (p. 358):
  - Bit 13 = 1: Hexapod strut / axis A / B in motion
  - Bit 13 = 0: Hexapod strut / axis A / B not in motion
  - Bit 15 = 1: Target position has been reached
  - Bit 15 = 0: Target position has not been reached

  You can query the register bits with the SRG? command (p. 247). In addition, you can record these bits with the C-887's data recorder (p. 97), record option 80 (status register of axis).
- Common system status register (p. 359):
  - Bits 8 to 15 = 1: Target position has not been reached (corresponding hexapod strut(s) / axis A / B in motion)
  - Bits 8 to 15 = 0: Target position has been reached (corresponding hexapod strut(s) / axis A / B not in motion)

  You can query the bits of the system status register with the STA? (p. 251) and #4 (p. 143) commands.

Furthermore, following commands are available for axis-related queries:

- #5 command (p. 143) command: Requests motion status of the axes.

  The motion status of *all* hexapod axes (X to W) is "in motion" as long as at least *one* hexapod strut is in motion (strut status according to the register bits stated above).
- ONT? command (p. 236) command: Queries the on-target state of the axis in question.

  The ONT? query only checks whether the end of the dynamics profile for the axes of the hexapod's motion platform (X to W) has been reached but **not** the actual motion status.

Settling window, settling time

The motion status of the hexapod struts refers to the target positions that the C-887 has calculated for the struts from the target positions of the hexapod axes. The C-887 determines the motion status of the hexapod struts and axes A and B using the following criteria:

- Settling window around the target position:

  Specification with ***Settling Window (encoder counts)*** parameter, ID 0x36. The parameter value corresponds to half of the window width. Unit: Counts of the encoder.
- Delay time for setting the status bit to "Target position has been reached":

  Specification with ***Settle Time*** parameter, ID 0x38. Unit: Number of servo cycles.
- The target position is considered as reached when the current position is inside the settling window and stays there at least for the duration of the delay time.

3.10 Commanding via the EtherCAT Interface

3.10.1 Introduction

The EtherCAT master specifies the target positions for the axes of the hexapod motion platform in Cartesian coordinates and evaluates the corresponding position feedback of the axes. The C-887.53x (.53x signifies C-887.53, .531, .532, .533) converts the axes' target positions to motion commands for the six hexapod drives. The hexapod system (C-887.53x plus hexapod) acts like an intelligent multi-axis drive pursuant to the CiA402 drive profile.

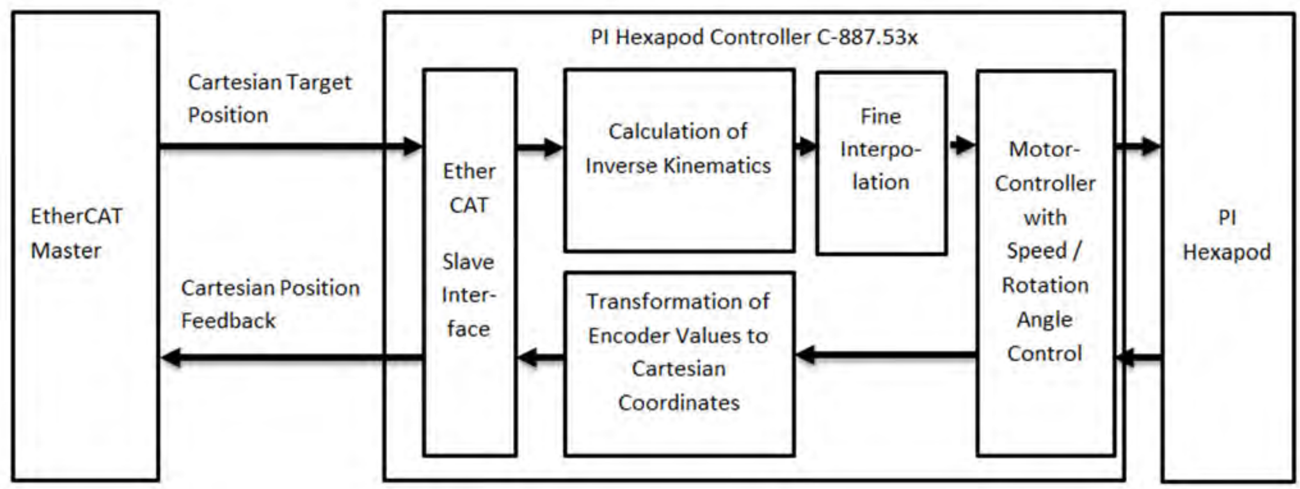


Figure 4: Hexapod system commanded from the EtherCAT master

The following table gives an overview of the most important characteristics for commanding the C-887.53x via an EtherCAT master:

Fieldbus protocol	EtherCAT (CoE = CANopen over EtherCAT)
Drive profile	CiA402 (IEC 61800-7-201)
Type of commanded device, number of axes:	Multi-axis device with 6 individual Cartesian axes
Cycle time for the specification of the target positions, signal processing, and synchronization:	≥1 ms

Operating modes supported according to CiA402:	**Mode** / **Object 0x6060** / **Note** No mode changes / no mode selected / 0 / Safe basic state, target positions will be ignored, required for activating coordinate systems Homing mode / 6 / Do a reference move Cyclic Synchronous Position Mode (CSP) / 8 / Cyclic specification of target positions by the EtherCAT master
Supported synchronization mode:	Distributed clocks, synchronous with SYNC0 event
Fieldbus connector:	RJ45 socket

The EtherCAT interface will be activated in the C-887.53x by a parameter setting, default setting: After switching on or rebooting the C-887.53x, the EtherCAT interface is activated. For details, see "Configuring the C-887 for Commanding by the EtherCAT Master" (p. 45).

Integration of the C-887.53x in the EtherCAT network is done via an XML file provided by PI; for details, see "Configuring the EtherCAT Master" (p. 46).

The **RUN** LED of the C-887.53x (p. 16) shows the current state of the EtherCAT communication state machine.

You can find further information in the "EtherCAT Interface Description" document (C887T0011).

3.10.2 Configuring the C-887 for Commanding by the EtherCAT Master

Activating the EtherCAT interface

In the case of models with EtherCAT interface, motion in the X, Y, U, V, W, and Z axes can be triggered by an EtherCAT master (for details, see "Motion of the Hexapod" (p. 31)). The EtherCAT interface of the C-887 is activated by the value of the ***Configure Command Mode*** parameter (ID 0x19002000). Possible values of the ***Configure Command Mode*** parameter:

- 0 = EtherCAT interface deactivated ("GCS")
- 1 = EtherCAT interface activated ("External: EtherCAT"); default setting for the C-887.53, .531, .532, .533 models.

The value of the ***Configure Command Mode*** parameter can be changed with GCS commands via the TCP/IP or RS-232 interface of the C-887.53x.

1. Select desired activation status:

```
SPA 1 0x19002000 1
```

or

```
SPA 1 0x19002000 0
```

2. Activate parameter value changes:
   a) Save parameter value to the nonvolatile memory of the C-887:
      `WPA 101 1 0x19002000`
   b) Send `RBT` to reboot the C-887 or switch the C-887 off and on.

EtherCAT interface and GCS commands

If the EtherCAT interface is activated by the ***Configure Command Mode*** parameter, it is still possible to send GCS commands via the TCP/IP or RS-232 interface of the C-887.53x. However, execution of the following GCS commands will be denied by the C-887.53x if the hexapod axes are in the ***Operation enabled*** state of the CiA402 state machine:

- Motion commands (e.g., MOV)
- Motion stop commands (e.g., HLT and STP)
- WPA command

The ***Operation enabled*** state cannot be exited via a GCS command.

If the EtherCAT interface is activated, the coordinate systems used (p. 38) are defined and activated by PI-specific objects (SDO). The settings for coordinate systems cannot be changed with GCS commands when the EtherCAT interface is activated.

GCS communication with the C-887.53x can reduce the performance of the EtherCAT interface. GCS communication should therefore be avoided (e.g., sending GCS commands or using PC software such as PIMikroMove) when using the EtherCAT interface. Use GCS communication only for initial startup, support cases, or for troubleshooting.

An inactive EtherCAT interface does not allow commanding via the EtherCAT master, but only via GCS commands.

3.10.3 Configuring the EtherCAT Master

The steps for configuration, startup, and operation of the EtherCAT master depend on the device used. Details can be found in the documentation of your EtherCAT master.

For the integration of the C-887 in the EtherCAT network, the .XML file "Physik_Instrumente_Hexapod.xml" provided by PI must be saved on the EtherCAT master. The XML file can be found in the EDS directory on the CD in the scope of delivery of the C-887.

In addition, the following settings of the EtherCAT master must be changed to adapt it to the C-887.

The following must be set for each Cartesian axis of the hexapod system:

- Scaling Factor Numerator: 1
  Scaling Factor Denominator: 100000
- Reference system: Absolute (the C-887 provides absolute positions to the EtherCAT master)
- Dead time compensation: depends on the cycle time. Calculation and examples:
  Dead time = 4 * Cycle time + 5 * Interpolation buffer time (2 ms)
  1 ms Cycle time: 14 ms Dead time

2 ms Cycle time: 18 ms Dead time
Empirical optimization recommended

- Acceleration, velocity, and jerk should be adapted to the hexapod to prevent motion errors (for technical data, see the documentation of the hexapod)

The following must be set for the entire hexapod system:

- Synchronization mode: Distributed clocks, synchronous with SYNC0 event

The minimum cycle time of the hexapod system (1 ms) must be maintained during commanding by the EtherCAT master. If the actual cycle time is shorter than the minimum cycle time of the hexapod system, the hexapod does not move.

During commanding by the EtherCAT master, the actual cycle time of the C-887 can be read out via the value of the ***Cycle Time For Interpolation In External Mode*** parameter (ID 0x19002010). The parameter value can be read out with the SPA? GCS command (p. 245) via the TCP/IP or RS-232 interface of the C-887.

3.11 Communication Interfaces

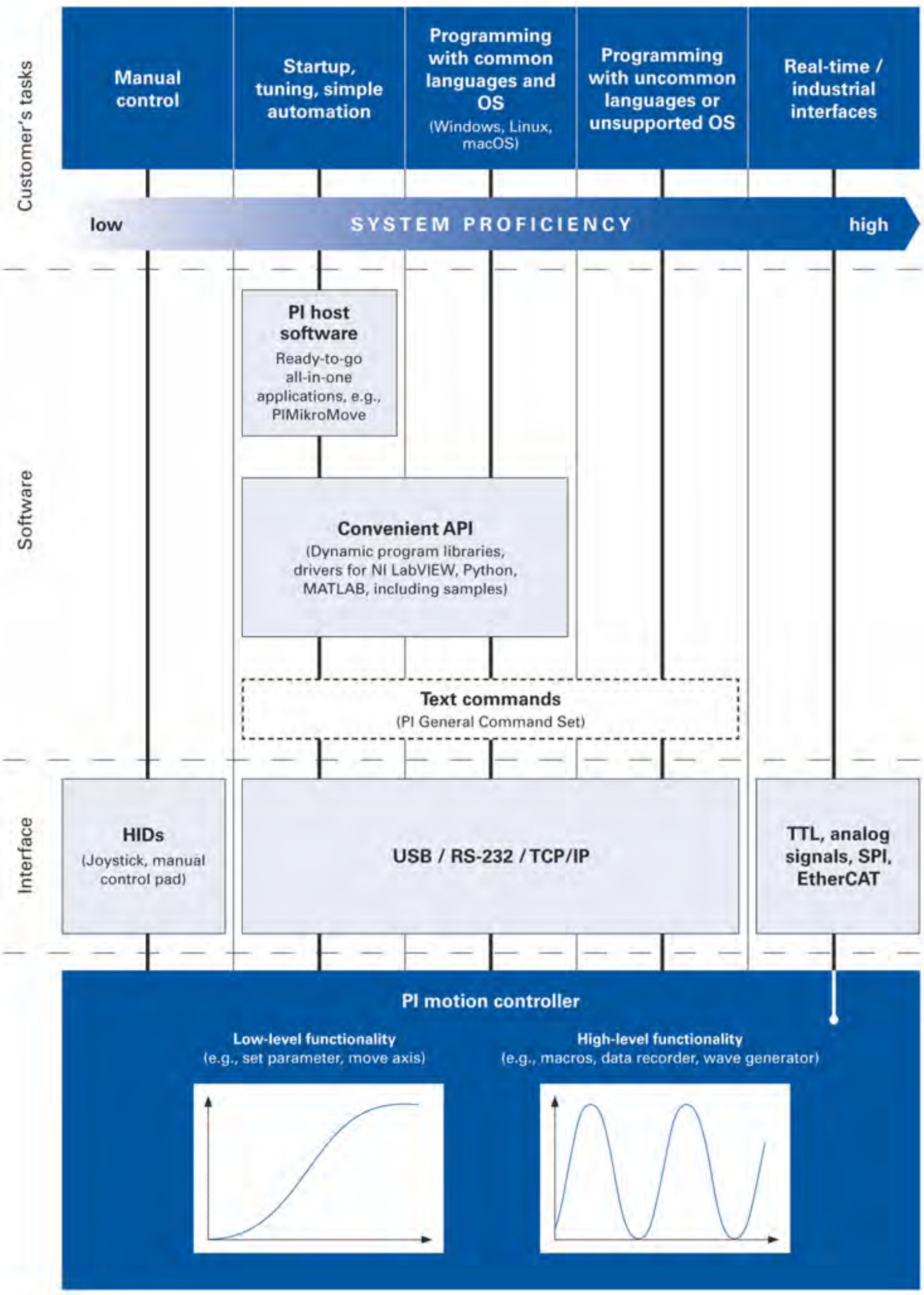
The C-887 can be controlled with ASCII commands via the following interfaces:

- TCP/IP
- Serial RS-232 connection

The C-887.53, .531, .532, and .533 models are equipped with an EtherCAT interface. When the hexapod system is commanded via the EtherCAT interface, it is used as a multi-axis device according to the CiA402 drive profile. The EtherCAT master commands the logical axes X, Y, Z, U, V, and W of the hexapod motion platform.

For further information, see "Commanding via the EtherCAT Interface" (p. 44).

3.12 Overview of PC Software



The following table shows a selection of PI software suitable for the C-887. The specified operating systems stand for the following versions:

- Windows: Versions 8.1, 10, 11 (32-bit, 64-bit)
- Linux: Kernel 4.15.0; glibc: 2.23; GTK2: 2.24.30

PC software	Supporting operating system	Short description	Recommended use
PIMikroMove	Windows	Graphical user interface for Windows with which the C-887 and other controllers from PI can be used: ▪ The system can be started without programming effort ▪ Graphic representation of the motion ▪ Macro functionality for storing command sequences on the PC (host macros) ▪ Complete environment for command entry, for trying out different commands No command knowledge is necessary to operate PIMikroMove.	For users who want to do simple automation tasks or test their equipment before or instead of programming an application. A log window showing the commands sent makes it possible to learn how to use the commands.
Dynamic program library for GCS PI GCS 2.0 DLL	Windows, Linux	Allows software programming for the C-887 with programming languages such as C++. The functions in the dynamic program library are based on the PI General Command Set (GCS).	For users who would like to use a dynamic program library for their application. Is required for PIMikroMove. Is required for the NI LabVIEW drivers.
PI Hexapod 3D Library	Windows	Program library that provides the following functions for PIMikroMove: ▪ 3-D display of hexapod motion and the active coordinate system ▪ Configuration and administration of user-defined coordinate systems	Is required for PIMikroMove.
Drivers for use with NI LabVIEW software	Windows	NI LabVIEW is a software for data acquisition and process control (must be ordered separately from National Instruments). The driver library is a collection of virtual instrument drivers for PI electronics. The drivers support the PI General Command Set.	For users who want to use NI LabVIEW to program their application.

PC software	Supporting operating system	Short description	Recommended use
MATLAB drivers	Windows	MATLAB is a development environment and programming language for numerical calculations (must be ordered separately from MathWorks). The PI MATLAB driver consists of a MATLAB class that can be included in any MATLAB script. This class supports the PI General Command Set. The PI MATLAB driver does not require any additional MATLAB toolboxes.	For users who want to use MATLAB to program their application.
PIPython	Windows, Linux	Collection of Python modules for convenient use of PI electronics and GCS data. The modules can be used with Python 3.6+. Use on other operating systems is also possible via sockets.	For users who want to use Python to program scripts for their application. The use of Python extends the scope of functions of the GCS commands considerably.
PITerminal	Windows, Linux	Terminal program that can be used for nearly all PI controllers (refer to the description of the ***Command Entry*** window in the PIMikroMove user manual).	For users who want to send GCS commands directly to the controller.
PIStages3Editor	Windows	Program for opening and editing positioner databases in .db format.	For users who want to deal with the contents of positioner databases more intensively.
PIUpdateFinder	Windows	Checks the PI software installed on the PC. If newer versions of the PC software are available on the PI server, they are offered for download.	For users who want to update the PC software.
PIHexapod SimulationTool	Windows	Simulation program with which the workspace and the load of the hexapod must be calculated before the hexapod is installed. Also recommended for testing user-defined coordinate systems.	For all users.
PIVirtualMove	Windows 10 or higher	Simulation program which can be used to simulate the workspace and load for hexapods and P-616 NanoCube® nanopositioners. Supports the handling of coordinate systems.	For all users.

PC software	Supporting operating system	Short description	Recommended use
PIHexapod Emulator	Windows	Program with which the C-887 and the connected hexapod as well as axes A and B can be emulated. The PIHexapodEmulator can be started directly (connect via TCP/IP with the address *localhost* and the port *50000*) or from PIMikroMove (select the C-887 in the ***Start up controller*** dialog and establish the connection on the ***PIHexapodEmulator*** tab).	For users who would like to test the behavior of the hexapod system when the controller and/or the hexapod are not available.
PIVeriMove	Windows	PIVeriMove hexapod software for collision checking Must be activated on the PC via license key. Refer to "Accessories" (p. 22) for further information.	For users who want to carry out collision tests.
PI Hexapod DataFiles	Windows	Configuration files for graphic display of the hexapods in the PC software	Required for the PI Hexapod 3D Library and PIVeriMove.
PIFirmware Manager	Windows	Program for updating the firmware of the C-887.	For users who want to update the firmware.

4 Unpacking

1. Unpack the C-887 with care.
2. Compare the contents with the scope of delivery according to the contract and the delivery note.
3. Inspect the contents for signs of damage. If any parts are damaged or missing, contact our customer service department immediately (p. 343).
4. Keep all packaging materials in case the product needs to be returned.

5 Installing

In this Chapter

5.1 General Notes on Installation

The hexapod can be mounted in any orientation.

NOTICE

Impermissible mechanical load and collisions!

Impermissible mechanical load and collisions between the hexapod, the load to be moved, and the surroundings can damage the hexapod.

- Only hold the hexapod by the base plate.
- Before installing the load, determine
  the limit value for the load and
  the workspace of the hexapod
  with a simulation program (p. 60).
- Avoid high forces and torques on the motion platform during installation.
- To avoid unintentional deactivation of the hexapod system and resulting position changes of the hexapod system, make sure that the power supply is not interrupted.
- Make sure that no collisions between the hexapod, the load to be moved, and the surroundings are possible in the workspace of the hexapod.

5.2 Installing the PC Software

The communication between the C-887 and a PC is necessary to configure the C-887 and send motion commands using the commands of the GCS. Various PC software applications are available for this purpose.

5.2.1 Doing Initial Installation

Accessories

- PC with Windows (8.1, 10; 32-bit, 64-bit) or Linux operating system and at least 30 MB free storage space
- Data storage medium with software from PI (included in the scope of delivery)

Installing the PC software in Windows

1. Start the installation wizard by double-clicking ***PISoftwareSuite.exe*** in the installation directory (root directory on the CD).

   The ***InstallShield Wizard*** window opens for installing the PC software.
2. Follow the instructions on the screen.

   The PI software suite includes the following components:
   - Driver for use with NI LabVIEW software
   - Dynamic program library for GCS
   - PIMikroMove
   - PC software for updating the firmware of the C-887
   - PI Update Finder for updating the PC software
   - USB driver

Installing the PC Software in Linux

1. Unpack the tar archive from the /Linux directory on the PI software CD to a directory on your PC.
2. Open a terminal and go to the directory where you unpacked the tar archive.
3. Log in as superuser (root privileges).
4. Enter ./INSTALL to start the installation.
   Pay attention to lower and upper case when entering commands.
5. Follow the instructions on the screen.

You can select individual components for installation.

5.2.2 Installing Updates

PI is constantly improving the PC software.

➢ Always install the latest version of the PC software and the positioner database.

Requirements

✓ Active connection to the Internet.

✓ If your PC uses a Windows operating system:

– You have downloaded the PI Update Finder manual (A000T0028) from the PI website. The link is in the "A000T0081-Downloading Manuals from PI.pdf" in the \Manuals folder on the PI software CD.

Updating the PC software and PISTAGES3.DB in Windows

➢ Use the PI Update Finder:

– Follow the instructions in the PI Update Finder manual (A000T0028).

Updating the PC software in Linux

1. Open the website https://www.physikinstrumente.com/en/products/software-suite (https://www.physikinstrumente.com/en/products/software-suite).
2. Scroll down to ***Downloads***.
3. Click on ***ADD TO LIST +*** for the ***PI Software Suite C-990.CD1***
4. Click on ***REQUEST***
5. Fill out the download request form and send the request.

   The download link will then be sent to the email address entered.
6. Unpack the archive file on your PC to a separate installation directory.
7. Go to the ***linux*** subdirectory in the directory with the unpacked files.
8. Unpack the archive file in the ***linux*** directory by entering the command tar -xvpf <name of the archive file> on the console.
9. Log into the PC as superuser (root privileges).
10. Install the update.

INFORMATION

If software is missing in the ***Downloads*** area or problems occur with downloading:

➢ Contact our customer service department (p. 343).

Updating PISTAGES3.DB in Linux

1. Contact the customer service department (p. 343) to get the latest version of the PISTAGES3.DB positioner database.
2. Log into the PC as superuser (root privileges).
3. Install the update that you received from our customer service department onto your PC.

5.2.3 Installing Custom Positioner Databases

PI provides a CD with a custom positioner that has the following contents:

- Program Import PI CustomStage
- Custom positioner database with the parameter set for the positioner

In order for the parameter set to be selected in the PC software, it must first be inserted into the PIStages3 positioner database by the Import PI Custom Stage program.

➢ Install the custom positioner database by double-clicking the file ***Import_PI_CustomStage.exe*** in the root directory of the CD.

The parameter set from the custom positioner database is inserted into PIStages3.

If a message appears that installation of the custom positioner database failed:

a) Update the PIStages3 database on your PC, see "Installing Updates" (p. 57).

b) Repeat the installation of the custom positioner database.

5.3 Ensuring Ventilation

High temperatures can overheat the C-887.

➢ Set up the C-887 with a distance of at least 10 cm to the top side and at least 5 cm to the sides. If this is not possible, make sure that the area is cooled sufficiently.

➢ Ensure sufficient ventilation at the place of installation.

➢ Keep the ambient temperature to a non-critical level (5 °C to 40 °C).

5.4 Grounding the C-887

The C-887 is not grounded via the power supply connection.

- ➢ Connect the threaded pin with the protective earth conductor symbol (see figure) on the housing of the C-887 to the protective earth conductor.

5.5 Connecting the C-887 to the Power Supply

INFORMATION

When the hexapod is connected to the **24 V Out 7 A** socket of the controller, the supply voltage of the controller is also used for the hexapod.

C-887.522, .523, .532, .533 models only:

The **E-Stop** socket controls an internal relay with N/O contact that deactivates or activates the 24 V output for the hexapod (**24 V Out 7 A** socket).

Requirements

- ✓ The C-887 is switched off, i.e., the on/off switch is in the O position.
- ✓ The C-887 is installed near the power supply so that the power plug can be quickly and easily disconnected from the mains.

Tools and accessories

- 24 V wide input range power supply included (for line voltages between 100 and 240 V AC at 50 or 60 Hz)
- Power cord included
- Alternative: Sufficiently sized power cord

Connecting the C-887 to the power supply

1. Connect the power supply's M12 connector (f) to the C-887's **24 V In 8 A** connector.
2. Connect the power cord to the power adapter.
3. Connect the power adapter to the power socket with the power cord.

5.6 Installing the Hexapod

5.6.1 Determining the Workspace and the Permissible Load for the Hexapod

➢ Follow the instructions in the document "PI Hexapod Simulation Tool - Determining the Workspace and the Permissible Load for the Hexapod" (A000T0068).

INFORMATION

Limit value only valid when servo mode is switched on:

The limit value for the load of the hexapod determined by the simulation program is only valid when servo mode is switched on for the axes of the motion platform. The maximum holding force is based on self-locking of the actuators in the hexapod struts when servo mode is switched off and is lower than the limit value when servo mode is switched on (see manual for the hexapod).

INFORMATION

The limit value for the load of the hexapod varies depending on the following factors:

- Activation state of the servo mode
- Installation position of the hexapod
- Load to be moved: mass and position of the center of mass on the motion platform
- Forces and torques acting on the motion platform
- Positions and orientations to be approached by the motion platform during operation (translational and rotational coordinates)

INFORMATION

The optionally available PIVeriMove hexapod software for collision checking makes it possible to check mathematically for possible collisions between the hexapod, load, and surroundings. The use of the software is recommended when the hexapod is located in a limited installation space and/or operated with a spatially limiting load. For details on activation and configuration of PIVeriMove, see the C887T0002 technical note (in the scope of delivery of the software).

5.6.2 Grounding the Hexapod

➢ Follow the instructions in the manual for the hexapod.

5.6.3 Mounting the Hexapod on a Surface

➢ Follow the instructions in the manual for the hexapod.

5.6.4 Fixing the Load on the Hexapod

- Follow the instructions in the manual for the hexapod.

5.7 Connecting the Hexapod to the C-887 with the Cable Set

INFORMATION

When the hexapod is connected to the **24 V Out 7 A** socket of the controller, the supply voltage of the controller is also used for the hexapod.

C-887.522, .523, .532, .533 models only:

The **E-Stop** socket controls an internal relay with N/O contact that deactivates or activates the 24 V output for the hexapod (**24 V Out 7 A** socket).

INFORMATION

If a safety function according to valid standards is required:

- Define and implement suitable measures. Examples of possible measures:
  - If the hexapod is supplied with power via the C-887, interrupt the supply line with suitable safety technology.
  - Use an external power adapter with suitable safety technology for the hexapod.
  - Only C-887.522, .523, .532, .533 models: Connect suitable safety technology to the **E-Stop** socket. Refer to "Using the E-Stop Socket" (p. 91) for further information.

INFORMATION

As of serial number 121017873 the C-887 also supports hexapod microrobots that use the BiSS protocol for data transmission.

Connecting the hexapod to the C-887 with the cable set

- Follow the instructions in the manual for the hexapod.
- Only C-887.522, .523, .532, .533 models: Connect the **E-Stop** socket so that motion of the axes is possible. For details, see "Using the E-Stop Socket" (p. 91).

5.8 Connecting Positioners for Axes A and B

NOTICE

Damage if a wrong motor is connected!

Connecting a positioner with a stepper motor can cause irreparable damage to the C-887.

➢ Only connect positioners with DC motor and integrated motor drivers to sockets **A** and **B** of the C-887.

INFORMATION

The **Motor A** and **Motor B** sockets (D-sub 15 (f)) of the C-887 are intended for connecting positioners with DC motor and integrated motor drivers.

If you inform PI on the positioner types used before the hexapod system is delivered, PI will configure the C-887 according to your order, so that the corresponding positioner types are assigned to axes A and B of the C-887:

- If you order only one positioner, the corresponding type is assigned to axis A.
- If you order two positioners, the corresponding types are assigned to axes A and B in ascending alphabetical order, for example, M-403.1DG to A and M-403.2PD to B; M-414.1PD to A and M-511.DD1 to B.

Refer to "Operating Parameters for Axes A and B" (p. 30) for further information on assigning positioner types.

Requirements

✓ The C-887 is switched off, i.e., the on/off switch is in the O position.

✓ You have read and understood the user manual for the positioner.

✓ You have mounted the positioner according to the description in the corresponding user manual.

Tools and Accessories

- Positioner from PI equipped with a DC motor and PWM amplifier, available as an optional accessory (p. 22)

Connecting positioners for axes A and B to the C-887

1. Connect the positioners to the **Motor A** and **Motor B** sockets.
   – If known, pay attention to the C-887 settings for the positioner type assignment.

   The positioner at **Motor A** is commanded as axis A, and the positioner at **Motor B** is commanded as axis B.
2. Use the integrated screws to secure the connections against accidental disconnection.

5.9 Connecting Analog Signal Sources

All C-887 models are equipped with four analog inputs on the **I/O** socket (p. 354).

The C-887.521 523, .531, and .533 models are additionally equipped with two high-resolution analog inputs on the **Analog In 5** and **Analog In 6** BNC sockets.

Refer to "Analog Input Specifications" (p. 348).

Application possibilities of the analog input signals:

- Use in fast alignment routines (p. 3). You can find further information in the "Fast Multi-Channel Photonics Alignment (FMPA)" document (E712T0016).
- Use in scanning procedures:
  - Scanning procedures carried out by the C-887: see the AAP (p. 147), FIO (p. 165), FLM (p. 168), FLS (p. 171), FSA (p. 176), FSC (p. 180), FSM (p. 184) commands
  - Scanning procedures carried out by PIMikroMove (available in the ***Tools*** menu of PIMikroMove): refer to the PIMikroMove manual
- Use in macros:
  - Conditional running of the macro: see WAC (p. 264)
  - Conditional stopping of the macro: see MEX (p. 227)
  - Conditional jump of the macro pointer: see JRC (p. 199)
  - Copying the input state to a variable: see CPY (p. 152)

  Details and examples of macros are found in "Controller Macros" (p. 122).

INFORMATION

For dynamic and high-resolution applications such as, for example, fast alignment routines or scanning procedures, it is recommended to use the **Analog In 5** and **Analog In 6** BNC sockets.
The analog inputs on the **I/O** socket are suitable for use in macros.

Tools and accessories

- Suitable signal source according to the input specifications (p. 348), for example, F-712.PM1 photometer (available as an optional accessory (p. 22)) for connecting to **Analog In 5** or **Analog In 6**

Connecting an analog signal source

1. Connect the signal source to the analog input.
2. Secure the connectors against accidental disconnection.

5.10 Connecting Digital Inputs and Outputs

The digital inputs and outputs on the **I/O** socket (p. 354) of the C-887 (TTL signals) can be used as follows:

- Outputs: Set the status with the DIO command (p. 155)
- Inputs: Get the status with the DIO? command (p. 156), used in macros:
  - Conditional running of the macro: see WAC (p. 264)
  - Conditional stopping of the macro: see MEX (p. 227)
  - Conditional jump of the macro pointer: see JRC (p. 199)
  - Copying the input state to a variable: see CPY (p. 152)

  Details and examples of macros are found in "Controller Macros" (p. 122).

Tools and accessories

- Suitable HD D-sub 26 connector (m)
- If you want to use the digital inputs: Suitable signal source according to the specifications of the input (p. 354)
- If you want to use the digital outputs: Suitable device for signal evaluation in accordance with the specifications of the output (p. 354)

Connecting digital inputs and outputs

➢ Digital inputs: Connect the signal source to one of the pins 7, 16, 17, and 25 of the **I/O** socket.

➢ Digital outputs: Connect the device for the signal evaluation to one of the pins 8, 9, 18, and 26 of the **I/O** socket.

➢ Secure the connectors against accidental disconnection.

5.11 Connecting the PC

The communication between the C-887 and a PC can be used to configure the C-887 and send motion commands with the commands of the GCS. The C-887 has the following interfaces for this purpose:

- TCP/IP
- RS-232 interface

In this section, you learn how to establish the corresponding cable connections between the C-887 and a PC. All further steps that are necessary for establishing communication between the C-887 and a PC can be found in "Establishing Communication via a TCP/IP Interface" (p. 72) and in "Establishing Communication via an RS-232 Interface" (p. 78).

5.11.1 Connecting the C-887 via the TCP/IP Interface

Requirements

- ✓ If the C-887 is to be directly connected to the PC:
  The PC has an unused RJ45 Ethernet connection socket.
- ✓ If the C-887 and a PC are to be operated together in a network:
  An access point to the network is available for the C-887; if necessary, a suitable hub or switch is connected to the network for this purpose.

Tools and accessories

- If the C-887 is to be connected directly to the PC:
  Crossover network cable (C-815.563 included in the scope of delivery)
- If the C-887 is to be connected to a network access point: Straight-through network cable (C-815.553 included in the scope of delivery)

Connecting the C-887 directly to the PC

➢ Connect the RJ45 socket of the C-887 to the RJ45 Ethernet socket of the PC using the crossover network cable.

Connecting the C-887 to the network in which the PC is also located

➢ Connect the RJ45 socket of the C-887 with the network access point using the straight-through network cable.

5.11.2 Connecting the C-887 via the RS-232 Interface

Requirements

- ✓ The PC has an unused RS-232 interface (also called a "serial interface" or "COM port", e. g. COM1 or COM2).

Tools and accessories

- RS-232 null-modem cable (C-815.34 included in the scope of delivery)

Connecting the C-887 to the PC

➢ Connect the **RS-232** panel plug of the C-887 and the RS-232 interface of the PC (one D-sub 9 panel plug (m)) with the null-modem cable.

5.12 Connecting the EtherCAT Master

The C-887.53x models (.53x stands for .53, 531, .532, and .533) are equipped with an EtherCAT interface.

Further information on the use of the EtherCAT interface can be found in the document "EtherCAT Interface Description" (C887T0011).

NOTICE

Malfunction due to incorrect connection!
Incorrect cable connections can lead to incorrect addressing and communication failure.

- ➢ Use the **Port 1** RJ45 socket (left) to connect the C-887.53x with the EtherCAT master (the **Port 2** RJ45 socket (right) is intended for connecting the next EtherCAT slave).
- ➢ Do **not** use EtherCAT and standard Ethernet together in a physical network. If possible, use cables with different colors for EtherCAT and standard Ethernet connections.

Tools and accessories

- Suitable cable:
  - CAT 5 patch cable or higher, straight-through or crossover
  - Cable length: 0.3 to 100 m

Connecting the EtherCAT master

- ➢ Connect the EtherCAT master with the **Port 1** RJ45 socket of the C-887.53x via a suitable cable.

6 Startup

In this Chapter

6.1 General Notes on Startup

CAUTION

Risk of crushing by moving parts!

Risk of minor injuries from crushing between the moving parts of the hexapod and a stationary part or obstacle.

➢ Keep your fingers away from areas where they could be caught by moving parts.

NOTICE

Incorrect configuration of the controller!

The configuration data used by the controller (e.g., geometrical data and servo control parameters) must be adapted to the hexapod. If incorrect configuration data is used, the hexapod can be damaged by uncontrolled motion or collisions.

When the controller is switched on or rebooted, the configuration data is adapted using the data that is loaded from the ID chip (p. 29).

➢ Once you have established communication via TCP/IP (p. 72) or RS-232 (p. 78), send the `CST?` command. The response shows the hexapod, to which the controller is adapted.

➢ Only operate the hexapod with a controller whose configuration data is adapted to the hexapod.

NOTICE

Assignment of an incorrect positioner type!

Unsuitable parameter settings might be loaded if a wrong positioner type is assigned to axes A or B. Unsuitable parameter settings can cause damage to the positioner during startup and operation.

➢ Make sure that the assigned positioner type matches the positioner connected.

➢ Change parameter values for axes A and B only after careful consideration.

NOTICE

Damage due to collisions!

Collisions can damage the hexapod, the load to be moved, and the surroundings.

- ➢ Make sure that no collisions are possible between the hexapod, the load to be moved, and the surroundings in the workspace of the hexapod.
- ➢ Do not place any objects in areas where they can be caught by moving parts.
- ➢ Stop the motion immediately if a controller malfunction occurs.

NOTICE

Damage from transport safeguard that has not been removed!

Damage can occur to the hexapod if the transport safeguard (p. 53) of the hexapod has not been removed and a motion is commanded.

- ➢ Remove the transport safeguard before you start up the hexapod system.

NOTICE

Damage from unintentional position changes!

The limit value for the load of the hexapod determined by the simulation program only applies when servo mode is switched on (p. 60) for the axes of the motion platform. The maximum holding force is based on self-locking of the actuators in the hexapod struts when the servo mode is switched off and is lower than the limit value when the servo mode is switched on (see manual for the hexapod).

When the actual load of the hexapod exceeds the maximum holding force based on the self-locking of the actuators, unintentional position changes of the hexapod can occur in the following cases:

- Switching off the C-887
- Rebooting the C-887
- Switching the servo mode off for the axes of the motion platform of the hexapod, e. g., by using the **E-Stop** socket (p. 91)

As a result, collisions are possible between the hexapod, the load to be moved, and the surroundings. Collisions can damage the hexapod, the load to be moved or the surroundings.

- ➢ Make sure that the actual load of the motion platform of the hexapod does not exceed the maximum holding force based on the self-locking of the actuators before you switch off the servo mode, reboot or switch off the C-887.

NOTICE

Damage from collisions during the reference move!

The hexapod moves unpredictably during a reference move. A collision check or prevention does **not** take place, even if a configuration for preventing collisions was stored on the C-887 with the PIVeriMove hexapod software for collision checking. Soft limits that have been set for the motion platform of the hexapod with the NLM and PLM commands are ignored during the reference move.

As a result, collisions are possible between the hexapod, the load to be moved, and the surroundings. Collisions can damage the hexapod, the load to be moved, and the surroundings.

- ➢ Make sure that no collisions between the hexapod, the load to be moved, and the surroundings are possible during the reference move of the hexapod.
- ➢ Do not place any objects in areas where they can be caught by moving parts during the reference move.
- ➢ After a successful reference move, supply a command for the corresponding target position in order to return to the reference position (default: zero position) from any given position. Do **not** start a new reference move.

NOTICE

Damage from uncontrolled motion of the hexapod!

The velocity and acceleration of the motion platform of the hexapod are **not** specified by the C-887 in the following cases:

- Cyclic transfer of target positions (p. 36)
- The hexapod (axes X, Y, Z, U, V, W) is still moving while a new motion command is sent. The previous target position is thereby overwritten without the velocity and acceleration of the motion platform of the hexapod being recalculated.

The platform of the hexapod then moves on an undefined path. On this undefined path, collisions with the surroundings of the hexapod are possible. Collisions can damage the hexapod, the load to be moved, and the surroundings.

If you trigger motion through the cyclic transfer of target positions:

- ➢ Set only target positions whose distance from each other is maximally as large as the value of the ***Path Control Step Size*** parameter (ID 0x19001504).

If you command point-to-point motion with motion commands (p. 31):

- ➢ Avoid sending new target positions while the hexapod (axes X, Y, Z, U, V, W) is still moving.
- ➢ If new target positions have to be sent while the hexapod is still moving (axes X, Y, Z, U, V, W): Only use motion commands to set target positions that maximally deviate from the current position by the value of the ***Path Control Step Size*** parameter (ID 0x19001504).

INFORMATION

The optionally available PIVeriMove hexapod software for collision checking makes it possible to check mathematically for possible collisions between the hexapod, load, and surroundings. The use of the software is recommended when the hexapod is located in a limited installation space and/or operated with a spatially limiting load. For details on activation and configuration of PIVeriMove, see the C887T0002 technical note (in the scope of delivery of the software).

INFORMATION

Application notes for the ID chip detection:

- Before you replace the connected hexapod, save the current parameter values of the controller on the PC (p. 301).
- Connect the hexapod only when the controller is switched off.
- When the firmware has finished booting, send the `CST?` command (p. 154) to check whether the installed configuration has to be activated by rebooting the controller. A reboot is necessary when the response is "NOSTAGE". The controller can be rebooted with the `RBT` command (p. 239).
- Send the `ERR?` command (p. 164) to check whether the configuration was activated successfully. If the response to ERR? contains the error code 233 or 211, the configuration for the new hexapod is not in the controller (possible for example, for customized hexapods or new standard hexapods). Contact our customer service department (p. 343) in order to receive a suitable configuration file. For the installation of the new configuration file, see "Updating Firmware and Configuration Files" (p. 318).
- Send the `VER?` command (p. 261) to check the information for the hexapod type, serial number, and manufacturing date saved on the ID chip. Example for the response:
  `IDChip: H-811.F-2 SN123456789 20/1/2016`

INFORMATION

The communication between the C-887 and a PC can be used to configure the C-887 and send motion commands with the commands of the GCS.

- Communication is possible via the RS-232 interface without further settings.
- For communication via TCP/IP, it may be necessary to adjust the interface parameters accordingly once (p. 73).

INFORMATION

The communication interfaces of the C-887 (TCP/IP, RS-232) are active at the same time. Commands are executed in the order in which the complete command lines arrive. The simultaneous use of several communication interfaces can cause problems with the PC software, however.

- Always only use one interface of the C-887.

INFORMATION

When switched on or rebooted, the C-887 automatically switches the servo mode on for all axes. When the servo mode is switched off for the axes of the motion platform of the hexapod (X, Y, Z, U, V, W), it is automatically switched on when the reference move is started.

INFORMATION

For axes with incremental sensors, motion can only be commanded after a successful reference move (p. 174) (also referred to as "initialization").

The behavior of the axes of the hexapod after the reference move is determined by the ***Behaviour After Reference Move*** (ID 0x07030401) and ***Target For Motion After Reference Move*** parameters (ID 0x07030402). Depending on the parameter values, the axes of the platform can be moved automatically e.g., to a specified position after the reference move.

- Value of the parameter 0x07030401 = 0: The axis remains in the reference position after the reference move.
- Value of the parameter 0x07030401 = 1: After the reference move, the axis moves to the absolute target position, which is specified by parameter 0x07030402.

A reference move is not required for axes with absolute-measuring sensors. The use of the FRF command is still recommended for these axes, however. FRF does **not** start a reference move for axes with absolute-measuring sensors but sets the target positions to the current position values. The above-described parameter values also go into effect, so that the axes can be moved to a defined "initial position", for example.

6.2 Switching the C-887 On

Requirements

✓ You have read and understood the general notes on startup (p. 67).

✓ The C-887 has been installed properly (p. 55).

Switching the C-887 On

➢ Set the toggle switch on the front panel of the C-887 to the I position.

The C-887 starts the operating system and the firmware. The starting procedure lasts approx. 40 seconds; the beginning and end are each indicated by a signal tone. After the end of the starting procedure, the **PWR** and **STA** LEDs light up.

During the starting procedure, the C-887 does the following actions, among others:

- Switches the servo mode on for the axes of the motion platform of the hexapod
- Activates the settings stored in the nonvolatile memory
- Runs the startup macro, if available

6.3 Establishing Communication via the TCP/IP Interface

Adaptation of the interface parameters

Before communication is established, it can be necessary to adapt the factory settings of the interface parameters once. The following interface parameters for the TCP/IP communication can be adapted in the nonvolatile memory of the C-887 with the `IFS` command (p. 196):

Interface parameters	Factory setting	Note
Default IP address (IPADR)	192.168.1.28:50000	Allows definition of a static (i.e., fixed) address. This static address is **not** used when the C-887 is configured for assignment of an IP address by a DHCP server or AutoIP (default setting of the startup behavior for configuring the IP address). If the static address is to be used: ▪ The startup behavior must be changed so that the C-887 uses the IP address defined with “IPADR". ▪ The IP addresses and subnet masks of the C-887 and PC as well as of all other network devices must be compatible with each other. For details, see "Preparing the PC and C-887 for Using Static IP Addresses" (p. 73).
Startup behavior for configuring the IP address for TCP/IP communication (IPSTART)	DHCP or AutoIP is used to obtain the IP address	The IP address of the C-887 is obtained via DHCP or automatically configured using AutoIP with the default setting of the startup behavior. The default setting of the startup behavior only needs to be changed if the network devices are to use static addresses instead.
Subnet mask (IPMASK)	255.255.255.0	The default setting of the subnet mask may have to be changed if the network devices are to use static addresses. For details, see "Preparing the PC and C-887 for Using Static IP Addresses" (p. 73).

After switching on or rebooting the C-887

The starting procedure of the C-887 must be finished before the communication between the C-887 and PC can be established. The starting procedure takes around 40 seconds (second signal tone indicates the end).

When the IP address of the C-887 is assigned via DHCP or the IP addresses of the network devices are configured with AutoIP, it will take up to 2 minutes after the end of the starting procedure of the C-887 (p. 71) for communication to be possible via TCP/IP.

Connection of the network cable when the controller is switched on

The establishment of communication via TCP/IP can fail if the network cable was connected to the RJ45 socket of the C-887 while the C-887 was switched on.

➢ If the establishment of communication fails, switch the C-887 off and back on again while the network cable is plugged in.

Port setting

For communication with GCS commands, the C-887 has one unchangeable port (50000) available.

6.3.1 Preparing the PC and C-887 for Using Static IP Addresses

If a network does not have a DHCP server or the C-887 is connected directly to the Ethernet socket on the PC **and** static IP addresses are to be used, it is necessary to adapt the interface parameters as follows:

- Set the startup behavior for configuring the IP address of the C-887 so that a static address is used
- The IP addresses and subnet masks of the C-887 and PC as well as all other network devices must be compatible with each other

To adapt IP addresses and subnet masks, you can choose one of the two following options:

- Adapt the PC settings and if necessary the settings of other network devices. The settings of the C-887 remain unchanged.
- Adapt the settings of the C-887. The PC settings and if necessary the settings of other network devices remain unchanged.

Requirement

✓ You have established communication between the C-887 and the PC via RS-232 in order to determine the settings of the C-887 and to be able to change them if necessary (p. 78).

Determining the PC's IP address and subnet mask

1. Open the window on your PC appropriately so that the TCP/IP properties can be displayed and set. The necessary steps depend on the operating system used.

If your operating system distinguishes between Internet Protocol version 4 (TCP/IPv4) and version 6 (TCP/IPv6) (e.g., Windows 10), open the window for version 4.

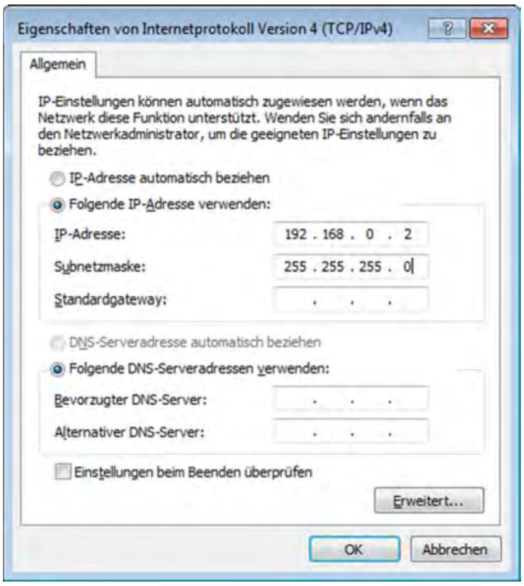


Figure 5: "Internet Protocol (TCP/IP) Properties" window with example settings (not necessarily suitable for your system)

The figure shows example settings that may not necessarily apply to your system.

2. Write down the settings.

Determining the IP address and subnet mask of the C-887, adapting the startup behavior of the C-887

1. If you have established communication between the C-887 and PC via the RS-232 interface, open the window for entering commands in the program used.
2. Enter the `IFS?` command.

   This command gets the values of the interface parameters in the nonvolatile memory.
3. Write down the settings ***IPMASK*** and ***IPADR***.
4. Make sure that the ***IPSTART*** parameter is correctly set:
   – If ***IPSTART*** = 0 (the static IP address defined with ***IPADR*** is used), the setting is correct.
   – If ***IPSTART*** ≠ 0: Send the command `IFS 100 IPSTART 0`.

INFORMATION

In PIMikroMove, you can determine the IP address and subnet mask of the C-887 and adapt the startup behavior of the C-887 in the ***Configure Interface*** window without sending commands.

Adapting IP settings of the PC

- If you want to leave the PC settings unchanged, continue with the section "Adapting C-887 settings" (p. 75).

1. Activate ***Use following IP address*** in the window in which the properties of the Internet protocol TCP/IP (TCP/IPv4) are displayed and set.
2. Adapt the IP address and the subnet mask to the settings of the C-887:
   a) Copy the first three sections of the IP address of the C-887 for the IP address on the PC.
   b) Make sure that the last section of the IP address on the PC differs from the last section of the IP address of the C-887 and is not "255" or "0".
   c) Copy the subnet mask of the C-887 for the subnet mask on the PC.

   Example:
   New IP address of the PC: 192.168.1.29 (if the C-887 has the IP address 192.168.1.28)
   New subnet mask of the PC: 255.255.255.0 (if the C-887 has the subnet mask 255.255.255.0)
3. Confirm the settings with the ***OK*** button.
4. If further network devices have to be adapted:
   Adapt the IP addresses and subnet masks as in the previous steps.
   Assign a separate, unique IP address to each network device.
   IP addresses must not occur twice in the same network.
5. Close the connection via the RS-232 interface, e. g., in PIMikroMove via the ***Connections > Close > C-887*** menu item in the main window.
6. Switch off the C-887.
7. Continue with the section "Establishing Communication via TCP/IP in the PC Software" (p. 76).

Adapting C-887 settings

1. Adapt the settings of the C-887 to those of the PC with the `IFS` command:
   a) Change the subnet mask with the `IFS 100 IPMASK xxx.xxx.xxx.xxx` command, whereby *xxx.xxx.xxx.xxx* is the subnet mask of the PC.
   b) Change the IP address with the `IFS 100 IPADR xxx.xxx.xxx.yyy:50000` command, whereby the following applies:
   – *xxx.xxx.xxx.* matches the first three sections of the IP address of the PC
   – *yyy* differs from the last section of the IP address of the PC and every other device in the same network
   – *yyy* is not "255" and not "0" and is in the address range that is given by the last section of the subnet mask
   – The port address "50000" must not be changed

Example:
If the IP address of the PC is 192.168.0.1 and no other device has the IP address 192.168.0.2, send the `IFS 100 IPADR 192.168.0.2:50000` command.

2. Close the connection via the RS-232 interface, e. g., in PIMikroMove via the ***Connections > Close > C-887*** menu item in the main window.
3. Switch off the C-887.
4. Continue with the section "Establishing Communication via TCP/IP in the PC Software" (p. 76).

6.3.2 Establishing Communication via TCP/IP in the PC Software

CAUTION

Risk of crushing from unexpected motion

When communication between the C-887 and the PC has been established via TCP/IP, the PC software offers all controllers present in the same network for selection. After a C-887 has been selected for the connection, all commands are sent to this controller. If the wrong controller is selected, unexpected motion of the hexapod could be commanded and cause minor crush injuries to the operating and maintenance staff.

- If several C-887 are displayed in the PC software, make sure that you select the right C-887.
- Assign a unique name affix for each C-887 in the same network: Enter the name affix as a value of the ***Customer Device Name*** parameter (ID 0x0D001000).

INFORMATION

In the list of controllers found in the same network, the C-887 is **not** shown as ***C-887*** but as ***HEXAPOD*** or ***F-HEX***, followed by the 9-digit serial number of the C-887.

A freely selectable name affix can be assigned for the C-887, in order to distinguish among several C-887 in the same network or on the same PC. Proceed as follows for assigning the name affix:

1. Establish communication via one of the available interfaces.
2. Change to command level 1 by sending:
   `CCL 1 advanced`
3. Enter a unique name affix as value of the ***Customer Device Name*** parameter (ID 0x0D001000) by sending:
   `SPA 1 0x0D001000` *name affix*
4. Save the new parameter value to the nonvolatile memory by sending:
   `WPA 101 1 0x0D001000`.
5. Close the connection.

Alternative:

- Identify the C-887 to be connected in the list of found controllers on the basis of its serial number (*SN*). The serial number of the controller is on the type plate on the rear panel of the C-887.

Requirements

- ✓ You have read and understood the general notes on startup (p. 67).
- ✓ The C-887 is connected to the network or directly to the PC via the RJ45 socket.
- ✓ If the C-887 is connected to a network:
  The PC to be used for communication with the C-887 is connected to the same network as the C-887.
- ✓ If the network does not have a DHCP server or if the C-887 is connected directly to the Ethernet connection socket on the PC **and** static IP addresses are to be used:
  By adapting the interface parameters, you have set the correct startup behavior for configuring the IP address of the C-887 and adapted the IP addresses and subnet masks of the C-887 and PC as well as all other network devices to each other (p. 73).
- ✓ If several C-887 are connected to the same network via their TCP/IP interfaces: You have assigned a unique name affix for the C-887 via the ***Customer Device Name*** parameter (ID 0x0D001000) in order to establish communication. Alternatively, you have the serial number of this C-887 ready. The serial number is on the type plate on the rear panel of the C-887.
- ✓ The PC is switched on.
- ✓ The required software is installed on the PC (p. 56).
- ✓ You have read and understood the manual for the PC software. The links to the software manuals are in the A000T0081 file on the PI software CD.
- ✓ The C-887 is switched off.

Establishing communication via TCP/IP

The procedure for PIMikroMove is described in the following. The procedure for the other PC software programs is similar (e.g., PITerminal, driver for use with NI LabVIEW software).

1. Switch the C-887 on.

   When the IP addresses of the network devices are configured with AutoIP, it will take up to 2 minutes after the end of the starting procedure of the C-887 (p. 71) until communication is possible via TCP/IP.

   – Wait until the AutoIP configuration has finished before you start PIMikroMove or another PC software from PI.

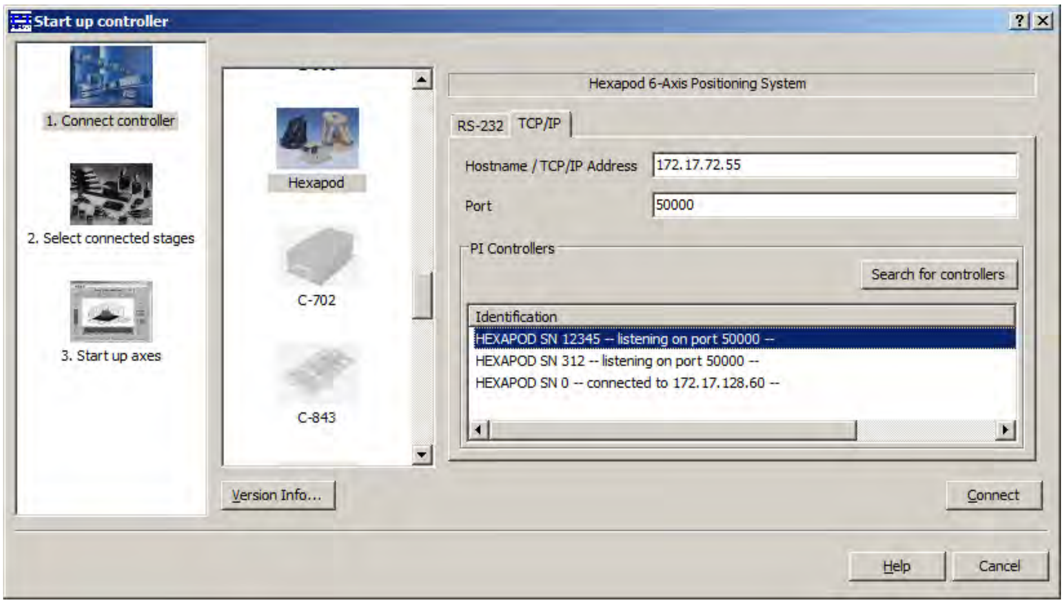
2. Start PIMikroMove.

   The ***Start up controller*** window opens with the ***Connect controller*** step.

   – If the ***Start up controller*** window does not open automatically, select the ***Connections > New...*** menu item in the main window.

3. Select ***C-887*** in the controller selection field.
4. Select the ***TCP/IP*** tab on the right-hand side of the window.

All controllers in the same network are shown.



5. Click the ***HEXAPOD SN ...*** or ***F-HEX SN ...*** entry in the list of controllers found (*SN* stands for *serial number*).
   - If ***HEXAPOD SN ...*** or ***F-HEX SN ...*** is displayed several times, identify your C-887 by the name affix that you previously assigned or by its nine-digit serial number (*SN* ...).

   If the C-887 is not displayed in the list of the controllers found:

   - The C-887 is in a different subnetwork than the PC. With the corresponding network configuration, you can still establish the communication if you know the current IP address of the C-887. Enter the IP address in the ***Hostname / TCP/IP Address*** field.
   - Check the network settings (p. 335). Consult your network administrator if necessary.
6. Check the IP address in the ***Hostname / TCP/IP Address*** field and the port number in the ***Port*** field.
7. Click the ***Connect*** button to establish communication.

When communication has been established successfully, the ***Start up controller*** window changes to the ***Start up axes*** step.

6.4 Establishing Communication via the RS-232 Interface

6.4.1 Changing the Baud Rate

The interface parameters for RS-232 communication are set at the factory as in the table below. The values in the nonvolatile memory can be queried with the `IFS?` command (p. 197).

Interface parameters	Factory setting	Note
Port for RS-232 communication (RSPORT)	1	Write-protected Indicates the port of the C-887 used for RS-232 communication.
Handshake for RS-232 communication (RSHSHK)	RTS/CTS	Write-protected Indicates the handshake setting of the C-887 for RS-232 communication.
Baud rate (RSBAUD)	115200	Indicates the baud rate of the C-887 for RS-232 communication. Further possible values are 9600, 19200, 38400, 57600. To establish communication successfully, the baud rates of the C-887 and PC must match.

Before communication is established, it can be necessary to change the factory baud rate setting of the C-887.

Requirements

- ✓ You have established the communication between the C-887 and the PC via one of the available interfaces.

Changing the baud rate for the RS-232 connection

1. Open the window for entering commands in the used program.
2. Send the `IFS 100 RSBAUD xxxxx` command, whereby *xxxxx* is the new baud rate.

   The baud rate setting is changed in the nonvolatile memory and is only effective after rebooting the C-887.

6.4.2 Establishing Communication via RS-232 in the PC Software

Requirements

- ✓ You have read and understood the general notes on startup (p. 67).
- ✓ The C-887 is connected to the RS-232 interface of the PC
- ✓ The C-887 is switched on and the starting procedure of the C-887 has finished (p. 71).
- ✓ The PC is switched on.
- ✓ The required software is installed on the PC.
- ✓ You have read and understood the manual for the PC software. The links to the software manuals are in the A000T0081 file on the PI software CD.

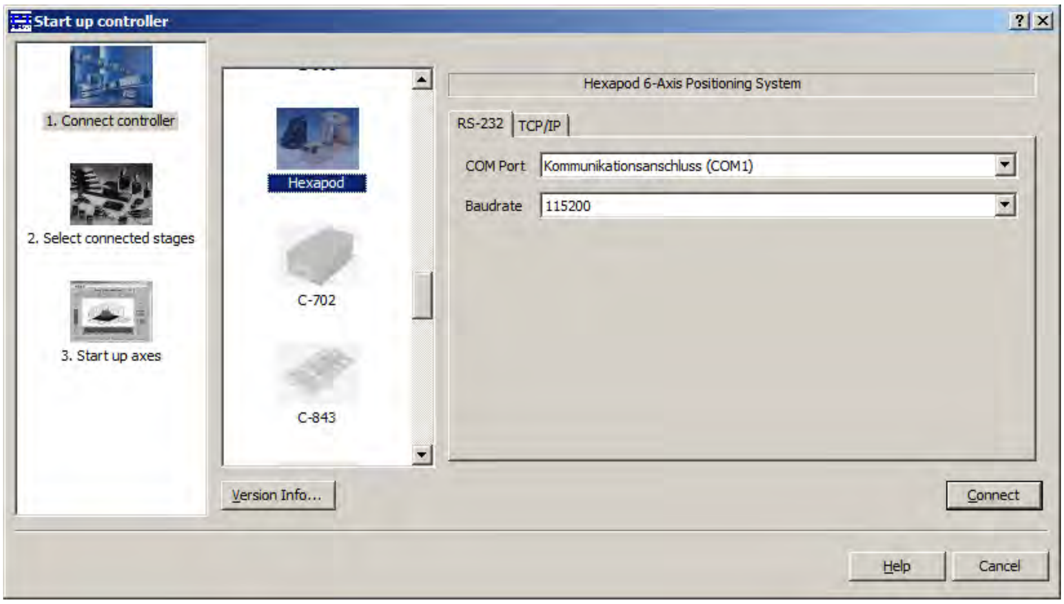
Establishing communication via RS-232

The procedure for PIMikroMove is described in the following. The procedure for the other PC software programs is similar (PITerminal, driver for use with NI LabVIEW software).

1. Start PIMikroMove.

   The ***Start up controller*** window opens with the ***Connect controller*** step.

   – If the ***Start up controller*** window does not open automatically, select the ***Connections > New...*** menu item in the main window.



2. Select ***C-887*** in the controller selection field.
3. Select the ***RS-232*** tab on the right-hand side of the window.
4. In the ***COM Port*** field, select the COM port of the PC to which you have connected the C-887.
5. In the ***Baud rate*** field, set the value 115200 (default setting on delivery of the C-887).

   This adapts the baud rate of the PC to the baud rate of the C-887.

   If you have changed the baud rate of the C-887, you have to enter the new value in the ***Baud rate*** field instead (p. 78).
6. Click ***Connect*** to establish communication.

When communication has been established successfully, the ***Start up controller*** window changes to the ***Start up axes*** step.

6.5 Starting Motion

In the following, PIMikroMove is used to move the axes.

Requirements

- ✓ You have read and understood the General Notes on Startup (p. 67).
- ✓ PIMikroMove is installed on the PC (p. 56).
- ✓ You have read and understood the PIMikroMove manual. The links to the software manuals are in the A000T0081 file on the PI software CD.
- ✓ You have read and understood the user manuals for all connected positioners (hexapod, axes A and B).
- ✓ You have installed the positioners according to the instructions in the corresponding user manuals and the C-887 is connected to them (p. 61, p. 62).
- ✓ You have established communication between the C-887 and the PC with PIMikroMove via the TCP/IP interface (p. 76) or the RS-232 interface (p. 78).

Starting motion with PIMikroMove

1. If necessary, assign the matching positioner type to axes A and B:

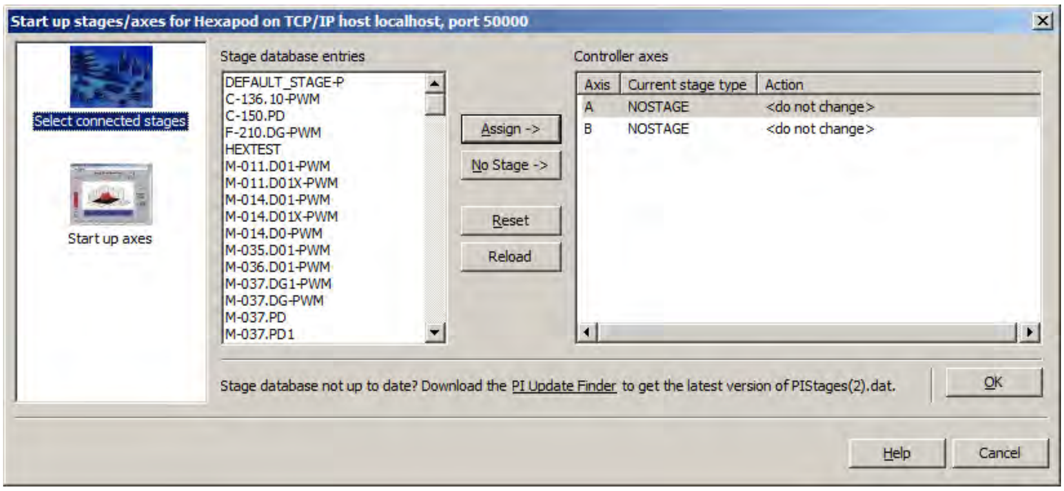
   a) Click ***2. Select connected stages*** on the left of the ***Start up controller*** window to switch to the step for assigning the positioner type.

   b) Mark the axis to which you want to assign a positioner type in the ***Controller axes*** list.

   c) Mark the positioner type in the ***Stage database entries*** list.

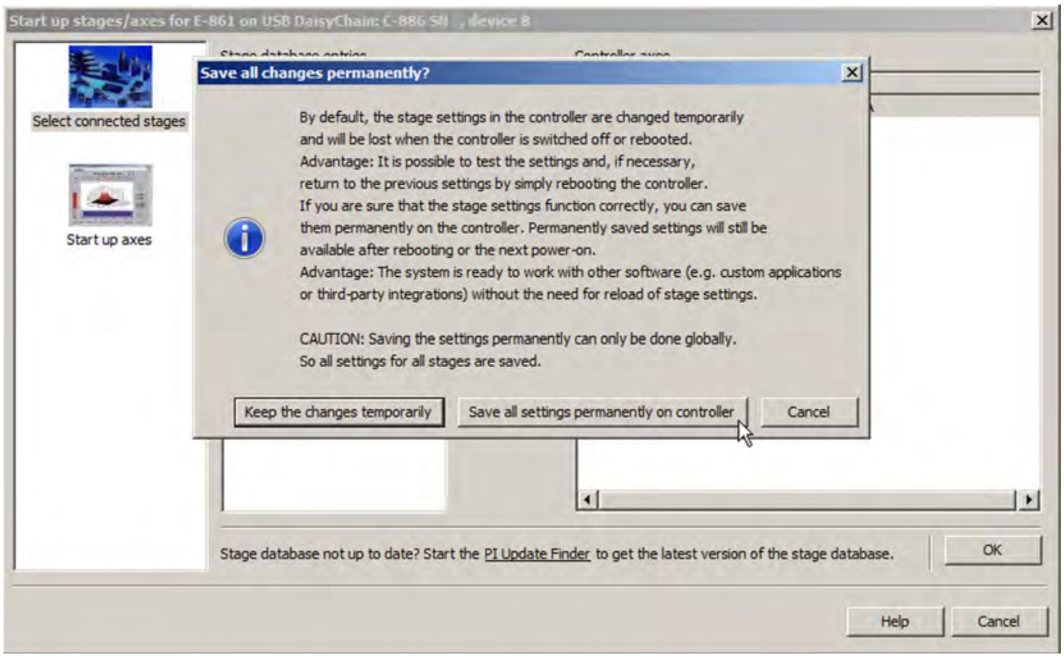
   d) Click ***Assign***.

   e) If necessary, repeat steps b to d for the second axis.



   f) Confirm selection with ***OK*** to load the parameter settings from the positioner database for the selected positioner type (for details, refer to "Operating

Parameters for Axes A and B" (p. 30)). The ***Save all changes permanently?*** dialog opens.

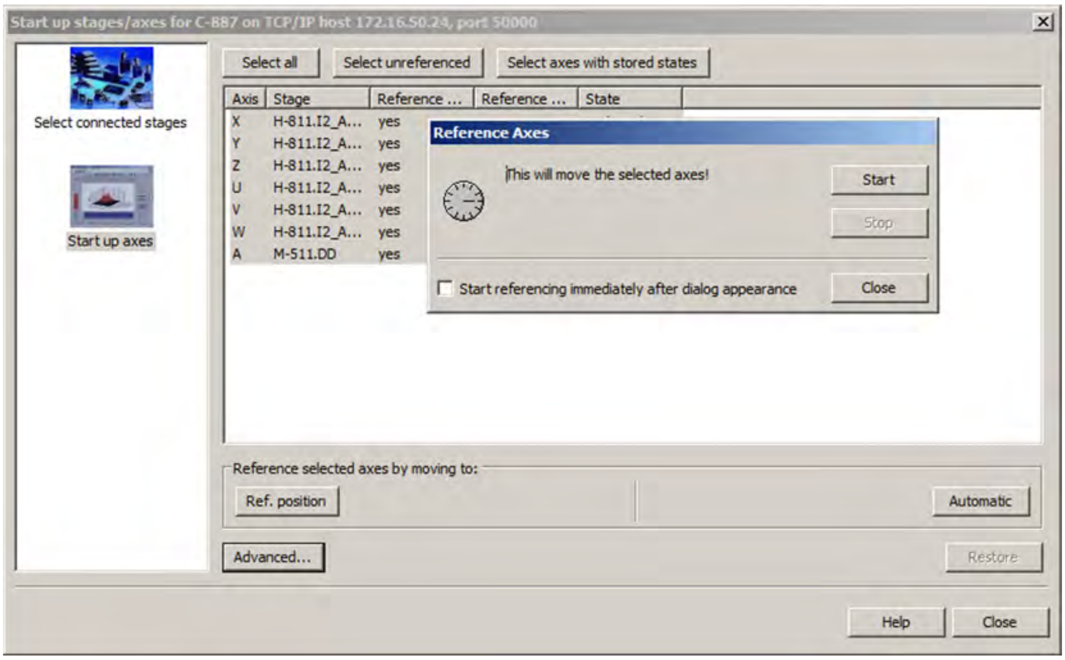


g) In the ***Save all changes permanently?*** dialog box, specify how you want to load the parameter settings:

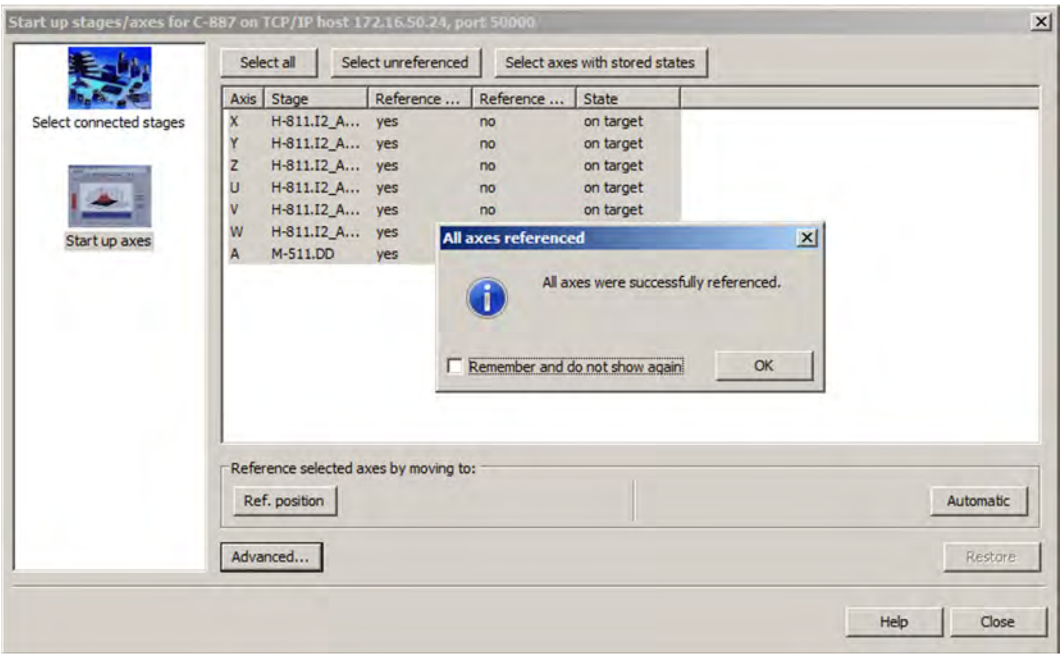
- Loading as default values (recommended): Click ***Save all settings permanently on controller*** to load the parameter settings into the nonvolatile memory. The settings are available immediately after switching on or rebooting the C-887 and do not need to be reloaded.
- Loading temporarily: Click ***Keep the changes temporarily*** to load the parameter settings into the volatile memory. The settings are lost when the C-887 is switched off or rebooted.

The ***Start up controller*** window changes to the ***Start up axes*** step.

2. Do the reference move for the axes in the ***Start up axes*** step (required for axes with incremental sensors). To do this, click ***Ref.position*** or ***Automatic***.



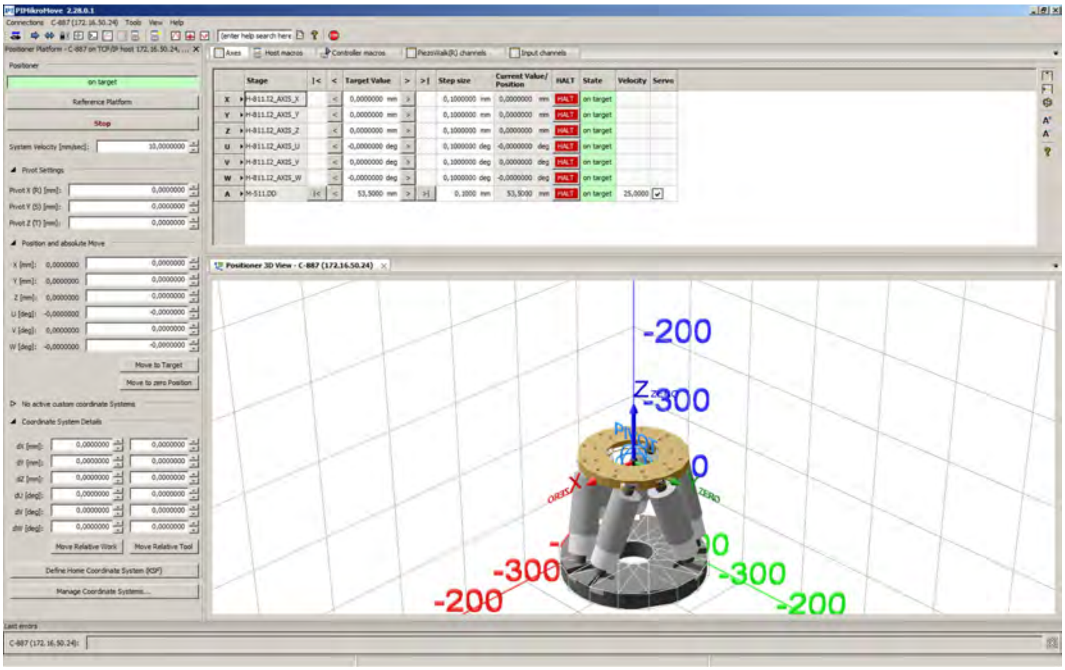
3. Click ***OK > Close*** after a successful reference move.



4. If the ***Positioner Platform*** window (left, docked) and the ***Positioner 3D View*** card are **not** displayed in the main window of PIMikroMove:

   – Display the ***Positioner Platform*** window with the ***C-887 > Show Positioner platform settings*** menu item.

- Display the ***Positioner 3D View*** panel with the ***C-887 > Positioner 3D View > Show*** menu item.
- If the ***Positioner 3D View*** card is not displayed in the foreground, click the corresponding tab.



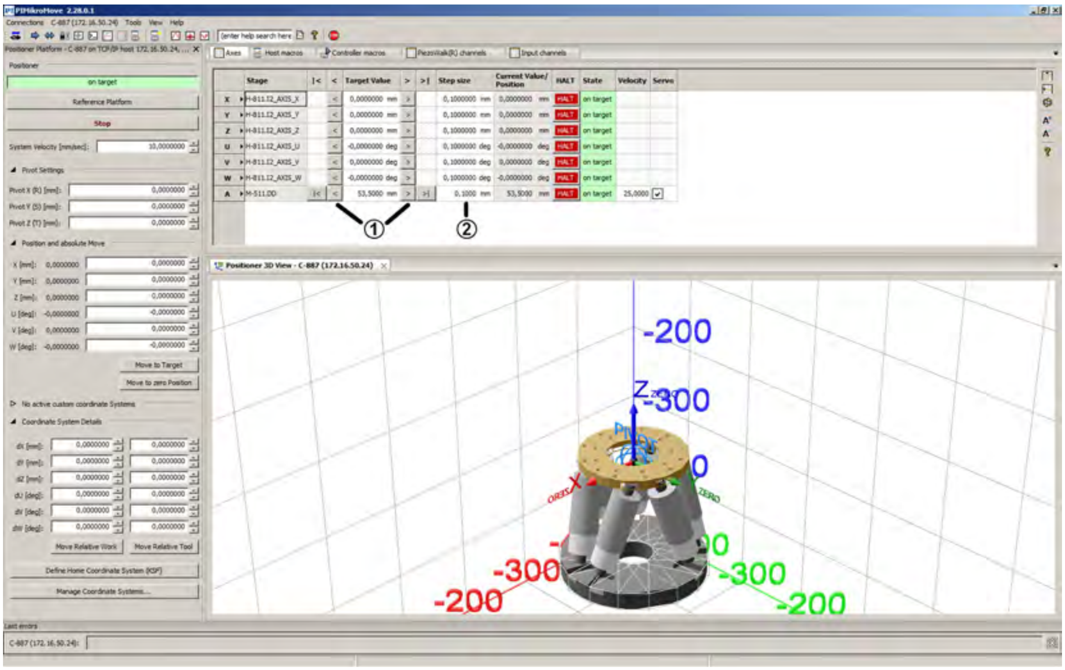
5. Test the motion for the axes of the hexapod's motion platform (X to W):
   a) Enter a target position for at least one axis of the motion platform of the hexapod in the ***Position and absolute Move*** area in the ***Positioner Platform*** window.
   b) Click ***Move to Target*** to start the motion towards the specified target position.
   c) When the motion is finished, repeat steps a and b for a new target position.

   If a node of the calculated dynamics profile or the target position cannot be reached, the motion will not be executed and a window will open with an error message.

   A new target position cannot be entered during motion.

   The motion is graphically displayed on the ***Positioner 3D View*** panel.
6. Start testing the motion of axes A and B on the ***Axes*** card in the main window of PIMikroMove.
   a) If necessary, switch servo mode on for axes A and B by activating the corresponding checkboxes in the ***Servo*** column.

b) Start test motion: You can for example, execute steps with a particular step size (2) by clicking the corresponding arrow keys (1) for an axis.



7 Operation

In this Chapter

7.1 General Notes on Operation

CAUTION

Risk of crushing by moving parts!

Risk of minor injuries from crushing between the moving parts of the hexapod and a stationary part or obstacle.

- Keep your fingers away from areas where they could be caught by moving parts.

NOTICE

Damage due to collisions!

Collisions can damage the hexapod, the load to be moved, and the surroundings.

- Make sure that no collisions are possible between the hexapod, the load to be moved, and the surroundings in the workspace of the hexapod.
- Do not place any objects in areas where they can be caught by moving parts.
- Stop the motion immediately if a controller malfunction occurs.

NOTICE

Damage from unintentional position changes!

The limit value for the load of the hexapod determined by the simulation program only applies when servo mode is switched on (p. 60) for the axes of the motion platform. The maximum holding force is based on self-locking of the actuators in the hexapod struts when the servo mode is switched off and is lower than the limit value when the servo mode is switched on (see manual for the hexapod).

When the actual load of the hexapod exceeds the maximum holding force based on the self-locking of the actuators, unintentional position changes of the hexapod can occur in the following cases:

- Switching off the C-887
- Rebooting the C-887
- Switching the servo mode off for the axes of the motion platform of the hexapod, e. g., by using the **E-Stop** socket (p. 91)

As a result, collisions are possible between the hexapod, the load to be moved, and the surroundings. Collisions can damage the hexapod, the load to be moved or the surroundings.

➢ Make sure that the actual load of the motion platform of the hexapod does not exceed the maximum holding force based on the self-locking of the actuators before you switch off the servo mode, reboot or switch off the C-887.

7.2 Protective Functions of the C-887

INFORMATION

If a safety function according to valid standards is required:

➢ Define and implement suitable measures. Examples of possible measures:
  - If the hexapod is supplied with power via the C-887, interrupt the supply line with suitable safety technology.
  - Use an external power adapter with suitable safety technology for the hexapod.
  - Only C-887.522, .523, .532, .533 models: Connect suitable safety technology to the **E-Stop** socket. Refer to "Using the E-Stop Socket" (p. 91) for further information.

7.2.1 Automatically Switching off the Servo Mode / Stopping the Motion

Axis motion is only possible when servo mode is switched on.

The C-887 automatically switches servo mode off for the hexapod axes (X, Y, Z, U, V, W) and axes A and B in the following cases, thereby stopping the motion:

Error code	Causes
66	The ***Check PowerGood Signal*** parameter (ID 0x19004000) has the value 1. Therefore, the C-887 checks the Power Good signal (pin 59 of the **Hexapod** socket). The check shows that the Power Good signal is outside of the required range (p. 356). Possible reasons: ▪ The power supply of the hexapod drives is interrupted or faulty. ▪ The connected hexapod is part of an older model series that is not equipped with a Power Good signal.
501	Only C-887.522, .523, .532, .533 models: As a result of the current connection of the **E-Stop** socket, the Power OK signal is **not** available, and the 24 V output for the hexapod (**24 V Out 7 A**) is deactivated. Refer to "Using the E-Stop Socket" (p. 91) for further information.
216	A limit switch is active for at least one hexapod strut or one of the axes A and B.

For certain errors, the handling of the axes can be configured. Depending on the configuration, the C-887 stops all axes or only the affected axis in the event of a fault, with or without switching off the servo mode.

The following parameters configure the handling of the axes:

Parameter	Description and Possible Values
Axis Handling on Motion Error (ID 0x19004001)	Selection of the axes that are handled when a fault occurs 0 = Handling only for the affected axis 1 = Handling of all axes (default)
Type of Axis Handling on Motion Error (ID 0x19004002)	Type of handling when fault occurs 0 = Stopping of the motion with STP (error code 10 is set), servo mode remains switched on 1 = Stopping the motion by switching off the servo mode (default)

The configuration becomes effective in the following cases:

Error code	Causes
1024	Motion error for at least one hexapod strut: Difference between current position and commanded position exceeds the maximum permissible value, for example, due to a drive or sensor malfunction or because the dynamic limits for the current load case were exceeded.
202	The motion was triggered by the wave generator (p. 99) or fast alignment routines (p. 3) and the internal fifo buffer no longer contains enough target positions.
657	The drive is blocked for at least one hexapod strut. Refer to "Configuring Safety Shutdown" (p. 95) for more information.
606	The maximum permissible velocity for error-free position detection was exceeded for at least one hexapod strut. Refer to "Configuring Safety Shutdown" (p. 95) for more information.

INFORMATION

Depending on the current velocity and load, the platform of the hexapod can move on an undefined path until a complete standstill after stopping.

The sensors of the hexapod are supplied with power via the data transmission cable (**Hexapod** socket (p. 356)). For this reason, the current position of the hexapod axes is still known when the power supply to the hexapod drives is interrupted.

Restoring operational readiness

1. Read out the error code and diagnosis information. Use the `ERR?` (p. 164) and `DIA?` (p. 154) commands or the corresponding operating elements of the PC software.

   The query resets the error code to zero.
2. Check your system and make sure that all axes can be moved safely. Refer also to "Troubleshooting" (p. 335).
3. If the power supply of the hexapod drives is interrupted: Restore the power supply.
4. Only C-887.522, .523, .532, .533 models: Connect the **E-Stop** socket so that motion of the axes is possible. For details, see "Using the E-Stop Socket" (p. 91).
5. Only if the connected hexapod does not have a Power Good signal (pin 59 of the **Hexapod** socket): Deactivate the Power Good signal check. For details, refer to "Troubleshooting" (p. 335).
6. Check the activation state of the servo mode: Use the `SVO?` command (p. 253) or the corresponding operating elements of the PC software.
7. If the servo mode is switched off: Switch on the servo mode for all axes. Use the `SVO` command (p. 252) or the corresponding operating elements of the PC software.

   A new reference move is not necessary.

7.2.2 Using the E-Stop Socket

The C-887.522, .523, .532, and .533 models has an **E-Stop** socket. The **E-Stop** socket controls an internal relay with N/O contact that activates or deactivates the 24 V output for the hexapod (**24 V Out 7 A** socket). An internal Power OK signal shows the C-887 the current connection of the **E-Stop** socket.

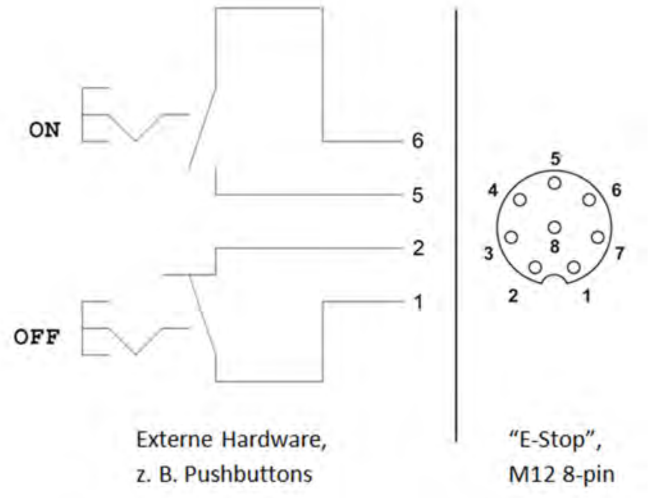


Figure 6: Connection of the E-Stop socket via external hardware

ON	Make contact
OFF	Break contact
E-Stop	Pins of the **E-Stop** socket on the front panel of the C-887, front view, for the complete pin assignment, see "E-Stop" (p. 354)

To make axis motion possible, pin 1 must be connected to pin 2 and pin 5 must be connected to pin 6 via the external connection of the **E-Stop** socket (24 V output of the C-887 is activated, the Power OK signal is provided).

If the connection between pins 1 and 2 and/or between pins 5 and 6 is interrupted, all axes are stopped. When the connection is interrupted, it is not possible to trigger motion again. Details:

- The internal Power OK signal is not available.
- The 24 V output of the C-887 is deactivated.
- The C-887 automatically switches servo mode off for the axes of the hexapod (X, Y, Z, U, V, W) and axes A and B and prevents switching on again.
- Error code 501 is set.
- When the hexapod is connected to the 24 V output of the C-887, the Power Good signal (pin 59 of the **Hexapod** socket) indicates that the power supply of the hexapod drives is interrupted (see also "Automatically Switching off the Servo Mode / Stopping the Motion" (p. 88)).

INFORMATION

The **E-Stop** socket **does not** offer a direct safety function in accordance with applicable standards (e.g., IEC 60204-1, IEC 61508, or IEC 62061). If a safety function according to valid standards is required, suitable measures must be defined and implemented by the operator of the hexapod system.

INFORMATION

Depending on the current velocity and load, the platform of the hexapod can move on an undefined path until a complete standstill after stopping.

The sensors of the hexapod are supplied with power via the data transmission cable (**Hexapod** socket (p. 356)). For this reason, the current position of the hexapod axes is still known when the power supply to the hexapod drives is interrupted.

Tools and accessories

If you want to use the **E-Stop** socket actively and require **no safety functions** according to valid standards:

- C-887.MSB motion stop button (available as an optional accessory (p. 22))

If you want to actively use the **E-Stop** socket and want to implement a safety function according to valid standards:

- Suitable external hardware for the connection of the **E-Stop** socket, for example, a pushbutton or switch
- If the power supply of the hexapod is not to take place via the 24 V output of the C-887: A separate suitable power adapter

If you do **not** want to use the **E-Stop** socket actively but want to bridge pin pairs 1/2 and 5/6 permanently (motion is always possible):

- C887B0038 shorting plug (included in the scope of delivery (p. 21))

Using the E-Stop socket

1. Connect the external hardware with the **E-Stop** socket according to the requirements of your application. The figures below show examples of connection variants.
   – Secure the connectors against accidental disconnection.
2. Make sure that the external wiring of the **E-Stop** socket allows axis motion during startup (p. 67) of the hexapod system.
3. If you have stopped the axes with the hardware connected to the **E-Stop**: Restore the operational readiness of the axes:
   a) Read out the error code. Use the `ERR?` command (p. 164) or the corresponding operating elements of the PC software.

      The error code 501 indicates that the connection to the **E-Stop** socket is preventing motion.

b) Check your system and make sure that all axes can be moved safely. Refer also to "Troubleshooting" (p. 335).

c) Restore the power supply of the hexapod drives.

d) Connect the **E-Stop** socket so that motion of the axes is possible.

e) Switch the servo mode on for all axes. Use the SVO command (p. 252) or the corresponding operating elements in the PC software.

A new reference move is not necessary.

Connection variant without safety function according to valid standards

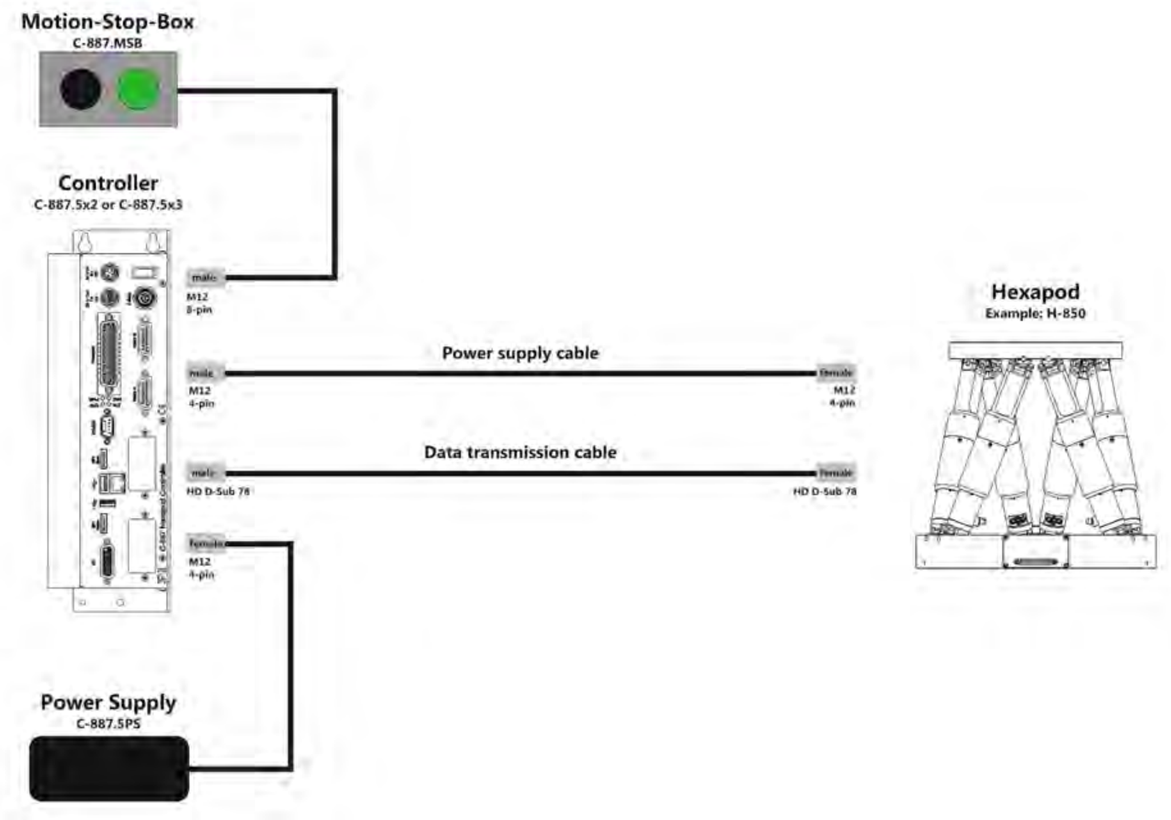


Figure 7: E-Stop socket with C-887.MSB motion stop box

Figure 8: Operating elements of the C-887.MSB motion stop box

STOP Mushroom head lock switch with locking
RUN Green pushbutton
Operation of the motion stop box:
Stopping the motion: Lock STOP by pressing
Restoring readiness for operation: Release STOP by pressing again, then press RUN

Connection variant with safety function

The hexapod is supplied with power via the C-887. The operator provides external hardware for the **E-Stop** socket with safety function according to valid standards.

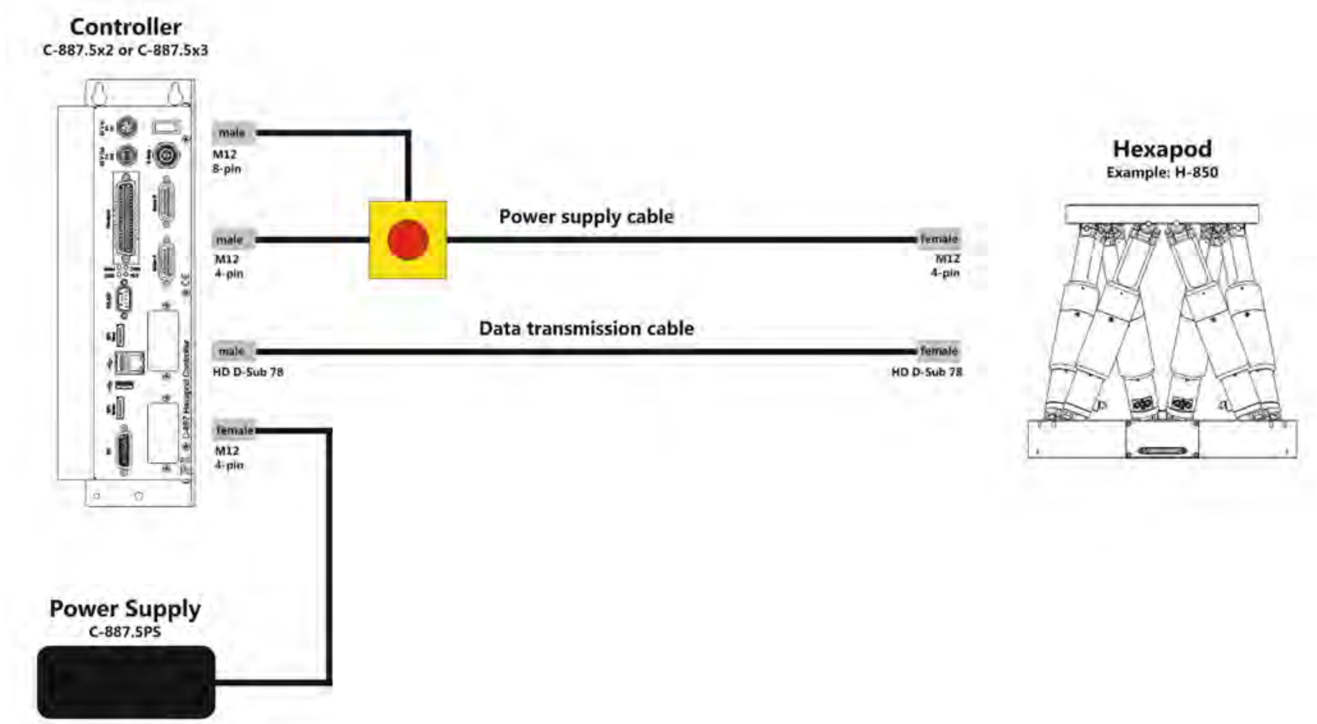


Figure 9: Using the E-Stop socket with external hardware

Connection variant with safety function

The operator provides external hardware for the **E-Stop** socket and a separate power adapter for the hexapod with a safety function according to valid standards.

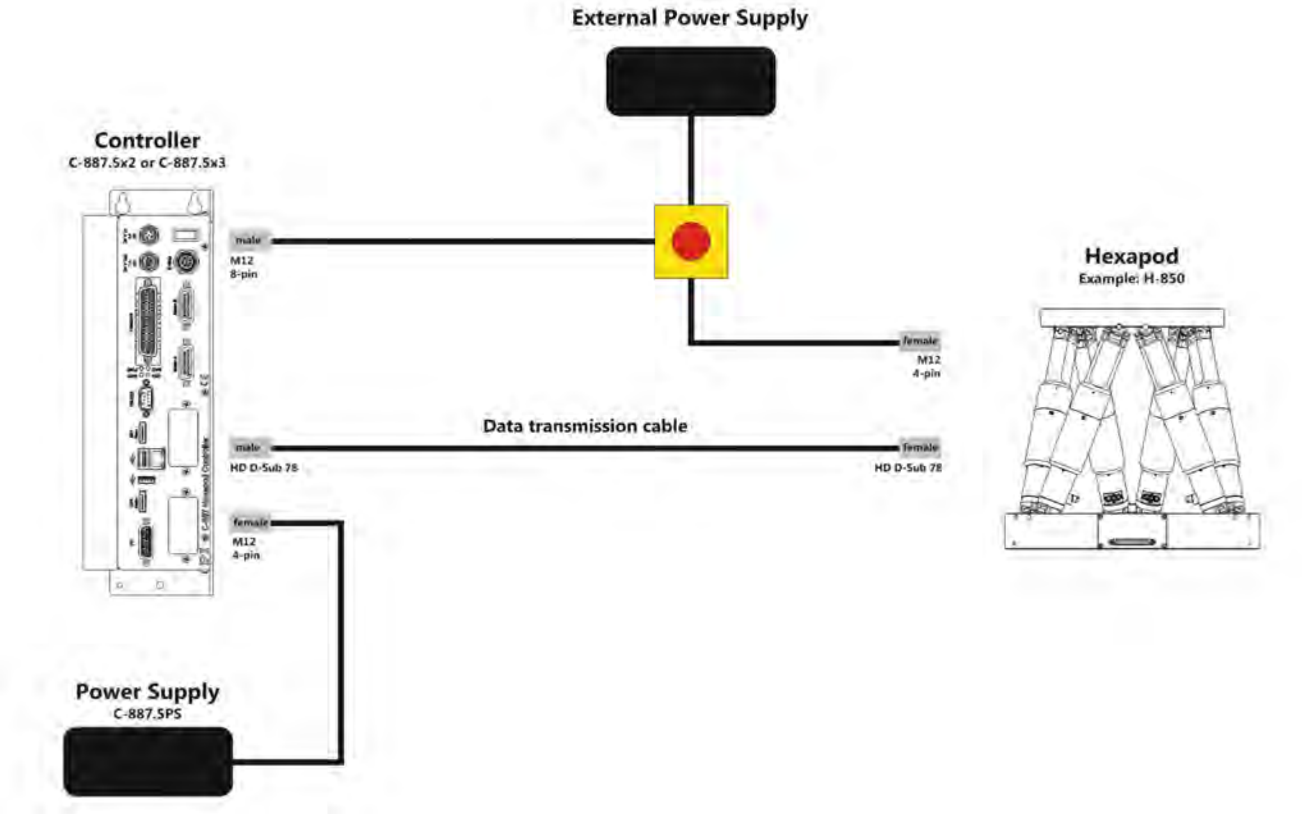


Figure 10: Using the E-Stop socket with external hardware, external power adapter for hexapod

7.2.3 Configuring Safety Shutdown

A safety shutdown can be configured in the C-887 for each hexapod strut. Depending on the configuration of the safety shutdown, the C-887 stops all axes or only the affected axis in the event of a fault, with or without switching off the servo mode.

Faults can be caused by the following, for example:

- Mechanical overload
- Excessive acceleration
- Faulty or incorrect sensor connector
- Electrical or mechanical sensor error
- Motor current limitation
- Phase error of the motor

The safety shutdown is based on the following checks:

- "Stall detection": Is the drive blocked?
- "Sensor error": Does the current velocity of the drive exceed the maximum permissible velocity for error-free position detection by the sensor?

For restoring operational readiness after a safety shutdown, see "Automatically Switching off the Servo Mode / Stopping the Motion" (p. 88).

The following parameters configure the safety shutdown:

Parameter	Description and Possible Values
Axis Handling on Motion Error (ID 0x19004001)	Selection of the axes that are handled when a fault occurs 0 = Handling only for the affected axis 1 = Handling of all axes (default)
Type of Axis Handling on Motion Error (ID 0x19004002)	Type of handling when fault occurs 0 = Stopping of the motion with STP (error code 10 is set), servo mode remains switched on 1 = Stopping the motion by switching off the servo mode (default)
Max Enc Vel 0x1A002000	"Sensor error" safety shutdown Maximum permissible velocity for error-free position detection by the sensor. If the velocity of the drive exceeds the value of the parameter, the safety shutdown is triggered and the error code 606 is set. in mm/s
Activate Motor Stuck Check (ID 0x1A002100)	"Stall detection" safety shutdown Status of the stall detection for the drive 0 = Stall detection deactivated (default) 1 = Stall detection activated For a meaningful use of the stall detection, the I term and the I limit must be set high enough for the position control of the drive. Sequence of the stall detection: ▪ Step 1: Check whether the motor control value of the drive has exceeded the value of the parameter 0x1A002200. ▪ Step 2: If motor control value > parameter 0x1A002200: After the delay time has expired (specified by parameter 0x1A002400), it is checked whether the velocity of the drive is greater than the value of the parameter 0x1A002300. ▪ Step 3: If velocity < parameter 0x1A002300, the drive is blocked. The safety shutdown is triggered and the error code 657 is set.
Minimal MotorOut to Move Axis (ID 0x1A002200)	"Stall detection" safety shutdown Motor control value, from which the drive has the "in motion" status dimensionless This parameter is only evaluated when parameter 0x1A002100 has the value 1.

Parameter	Description and Possible Values
Velocity Threshold under which Axis is considered not moving (ID 0x1A002300)	"Stall detection" safety shutdown Velocity from which the drive has the "in motion" status. Meaningful values depend on the sensor resolution and the minimum system velocity (parameter 0x19001501) of the hexapod. in mm/s This parameter is only evaluated when parameter 0x1A002100 has the value 1.
Time Period for which Axis is not yet considered not moving (ID 0x1A002400)	"Stall detection" safety shutdown Delay time for checking the velocity of the drive, applies to the entire system in s This parameter is only evaluated when parameter 0x1A002100 has the value 1.

7.3 Data Recorder

7.3.1 Data Recorder Properties

The C-887 contains a real-time data recorder. The data recorder can record different values for axes (e.g. current position) as well as the signals of the analog inputs.

The recorded data is temporarily stored in 16 data recorder tables with up to 262144 points each. Each data recorder table contains the data of one data source.

You can configure the data recorder for example, by defining the data type to be recorded and the data sources, and by specifying how the recording is to be started.

7.3.2 Configuring the Data Recorder

INFORMATION

The settings for configuring the data recorder can only be changed in the volatile memory of the C-887. After the C-887 has been switched on or rebooted, factory settings will be active unless a configuration is done with a startup macro.

Reading general information about the data recorder

- Send the HDR? command (p. 189).

The options available for recording and triggering are displayed together with the information on additional parameters and commands for data recording.

Configuring the data recorder

You can assign the data sources and record options to the data recorder tables.

- Send the DRC? command (p. 159) to read out the current configuration. Data recorder tables with the record option 0 are deactivated, i.e., nothing is recorded. By default, the data recorder tables of the C-887 record the following:
  - Data recorder table 1, 3, 5, 7, 9, 11: commanded position of the axes X, Y, Z, U, V and W
  - Data recorder table 2, 4, 6, 8, 10, 12: current position of the axes X, Y, Z, U, V and W
  - Data recorder table 13: the time
- Configure the data recorder with the DRC command (p. 157).

INFORMATION

Record option 80 (***Status register of axis***) can be used to record the bits of the signal status register (p. 358) for struts 1 to 6 of the hexapod and for axes A and B.

- To record the signal status register, use the ***Data Recorder*** window in PIMikroMove, which allows the targeted selection of single bits for the graphic display of the recorded data. For details, see the PIMikroMove manual.

You can specify how the recording is to be triggered.

- Get the current trigger option with DRT? (p. 163)
- Change the trigger option with the DRT command (p. 162). The trigger option applies to all data recorder tables whose record option is not set to 0.

You can specify the maximum number of points that are to be recorded for each data recorder table.

- Use the SPA command (p. 244) to change the ***Data Recorder Points Per Table*** parameter, ID 0x16000201. Maximum value: 262144 (default: 8192 points).

Setting the record table rate

- Send the RTR? command (p. 241) to read out the record table rate.
  The record table rate indicates via a factor the frequency with which data points are recorded. The possible range of values of the factors for RTR is 1 to 10000. The factor 1 corresponds to the frequency 10 kHz. The default value is 10 and corresponds to 1 kHz.
- Change the record table rate with the RTR command. (p. 240)

The greater the record table rate set with RTR, the greater the maximum duration of the data recording.

INFORMATION

When the C-887 runs fast alignment routines (p. 3), it sets the record table rate to the factor 4.

7.3.3 Starting the Recording

➢ Start the recording with the trigger option set with `DRT`.

Regardless of the trigger option set, the data recording is always triggered in the following cases:

- Start of a step response measurement with `STE` (p. 251)
- Start of an impulse response measurement with `IMP` (p. 198)
- Start of the wave generator output with `WGO` (p. 272)
- During wave generator output: `WGR` (p. 274) starts recording on the next output cycle of the wave generator

The data recording always takes place for all data recorder tables whose record option is not set to 0. It ends when the data recorder tables are full.

7.3.4 Reading Recorded Data

INFORMATION

Reading the recorded data can take some time, depending on the number of data points.
The data can also be read while data is being recorded.

➢ Read out the last recorded data with the `DRR?` command (p. 160).

The data is output in the GCS array format (refer to the SM146E user manual).

➢ Query the number of points in the last recording with the `DRL?` command (p. 160).

7.4 Wave Generator

7.4.1 Functionality of the Wave Generator

The wave generators are permanently assigned to the axes of the C-887; see "Commandable Elements" (p. 25).

A wave generator outputs absolute target positions for the axis motion on the basis of defined waveforms and thereby determines the dynamics profile. The wave generator output is especially suited to dynamic applications with periodic axis motion.

The following block diagram shows the integration of a wave generator in the C-887.

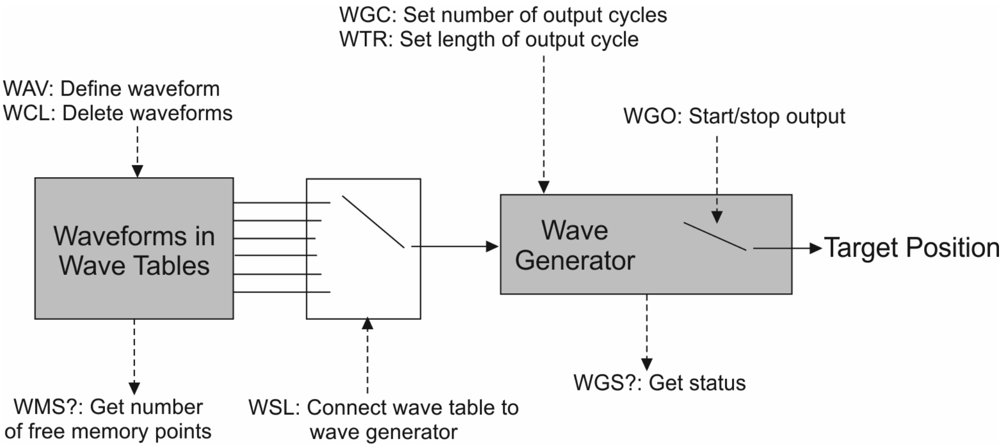


Figure 11: Block diagram of a wave generator

Waveforms can be defined and temporarily stored in up to 100 wave tables in the volatile memory of the C-887 (p. 103). Each wave table contains the data of one waveform. A total of 10.000.000 memory points are available for wave tables. The memory points are distributed to the individual wave tables when the waveforms are defined.

The wave tables can be assigned to the wave generators and therefore the axes as desired (p. 111). A wave table can be used by several wave generators at the same time.

The number of output cycles (p. 111) and the output rate (p. 111) of the wave generator can be set.

INFORMATION

You can permanently save the settings of the wave generator in the C-887 with the macro functionality of the C-887. You can also use a startup macro to configure the wave generator and start the output each time that the C-887 is switched on or rebooted. For details, see "Application Tips: Using Macros for the Wave Generator" (p. 120).

INFORMATION

It is recommended to use the PI Frequency Generator Tool or the PI Wave Generator Tool for work with the wave generator (both available in PIMikroMove).

No command knowledge is necessary to work with PIMikroMove. Nevertheless, it is recommended to familiarize yourself with the wave generator in this chapter.

All examples in this chapter can be entered in PIMikroMove or PITerminal as command sequences.

The use of the PI Frequency Generator Tool and the PI Wave Generator Tool is described in the PIMikroMove manual (SM148E).

You can use macros to define waveforms and configure wave generators. For work with macros, see "Controller Macros" (p. 122).

7.4.2 Commands and Parameters for the Wave Generator

Commands

The following commands are available for using the wave generator:

Command	Syntax	Function
GWD?	GWD? [<StartPoint> <NumberOfPoints> [{<WaveTableID>}]]	Gets the content of the wave tables (i.e., the waveforms).
TWG?	TWG?	Gets the number of wave generators (= number of axes).
WAV	WAV <WaveTableID> <AppendWave> <WaveType> <WaveTypeParameters>	Defines the waveform.
WAV?	WAV? [{<WaveTableID> <WaveParameterID>}]	Gets the current length of the wave tables (number of points).
WCL	WCL {<WaveTableID>}	Deletes the contents of the wave tables.
WGC	WGC {<WaveGenID> <Cycles>}	Sets the number of output cycles.
WGC?	WGC? [{<WaveGenID>}]	Gets the number of output cycles.
WGO	WGO {<WaveGenID> <StartMode>}	Starts and stops the output of the wave generator.
WGO?	WGO? [{<WaveGenID>}]	Gets the activation state that was last commanded for the wave generator.
WGR	WGR	Starts the data recording again while the wave generator is running.
WGS?	[<WaveGenID> [<InfoType>]]	Gets the status of the wave generator.
WMS?	[{<WaveTableID>}]	Gets the number of free memory points for the wave table.
WSL	WSL {<WaveGenID> <WaveTableID>}	Establishes the connection between the wave table and the wave generator.
WSL?	WSL? [{<WaveGenID>}]	Gets the connection between the wave table and the wave generator.
WTR	WTR {<WaveGenID> <WaveTableRate> <InterpolationType>}	Sets the table rate of the wave generator (thus influencing the duration of an output cycle).

Command	Syntax	Function
WTR?	WTR? [{<WaveGenID>}]	Gets the table rate of the wave generator.
#9	#9	Gets the current activation state of the wave generator.

Parameter

The following parameters define the characteristics of the wave generator:

Parameter	Description and Possible Values
Maximum Number Of Wave Points (ID 0x13000004)	Total number of available points for waveforms The wave tables of the C-887 have a total of 10.000.000 points. The available points are distributed among the wave tables when waveforms are defined with the WAV command (p. 265). This parameter is write-protected.
Number of Wave Tables (ID 0x1300010A)	Number of wave tables for saving waveforms The C-887 has 100 wave tables. This parameter is write-protected.
Start All Hexapod Wave Generators (ID 0x19002001)	Start behavior of the wave generators for the axes of the motion platform of the hexapod (X, Y, Z, U, V, W): 0 = Every wave generator that is to be started must be addressed in the WGO command (default setting). 1 = Starting a wave generator starts all wave generators that are connected with a wave table. This option exists only for compatibility reasons. For the axes of the motion platform of the hexapod whose wave generator has **not** been started, the last valid target position is always commanded.

7.4.3 Defining the Waveform

Waveforms are defined with the following steps:

- Optional: Getting information on wave tables (p. 103)
- Creating a waveform in a wave table (p. 103)
- Optional: Deleting the wave table content (p. 104)

This manual contains examples for creating waveforms (p. 105).

INFORMATION

With the macro functionality of the C-887, you can permanently save the contents of the wave tables (= defined waveforms) in the C-887. For details, see "Application Tips: Using Macros for the Wave Generator" (p. 120).

Optional: Getting information on wave tables

- Send the `SPA? 1 0x13000004` command to get the total number of points that the C-887 provides for defining waveforms in wave tables.
- Send the `SPA? 1 0x1300010A` command to get the number of wave tables available in the C-887.
- Get the current number of already defined waveform points for the wave tables with the `WAV?` command (p. 270).
- Get the current contents of the wave tables (= already defined waveforms) with the `GWD?` command (p. 188).

  The response contains the contents of the wave table in the GCS array format (see separate manual for GCS array, SM 146E).
- Get the number of available memory points for the wave tables with the `WMS?` command (p. 275).

Creating a waveform in a wave table

1. Make sure that the selected wave table is **not** connected to a wave generator for which the output has been started. For details see "Configuring a Wave Generator" (p. 111) and "Stopping the Wave Generator Output" (p. 114).
2. Create the waveform in the selected wave table from single segments with the `WAV` command (p. 265) (WAV + max. 12 arguments). Supported wave types:

   "PNT" (user-defined wave)

   "SIN_P" (inverted cosine wave)

   "RAMP" (ramp wave)

   "LIN" (wave in the form of a single scan line)

   The waveform is written to the selected wave table in the volatile memory. For details, see "Examples for creating waveforms" (p. 105).

INFORMATION

When a waveform is defined with `WAV` (p. 265), the target positions and the resulting velocities are **not** checked. The check is not performed until during the wave generator output (p. 113).

INFORMATION

When a waveform is defined with WAV, the C-887 does **not** check the target positions and the resulting velocities. The check is not performed until during the wave generator output (p. 113). If motion is not possible, the C-887 stops the wave generator output abruptly and therefore the motion.

To define suitable waveforms with WAV (p. 265) and prevent the wave generator output from aborting, pay attention to the following:

- Calculate the waveform externally as precisely as possible (e.g., with NI LabVIEW, MATLAB or Python) before you transfer the waveform definition to the C-887 with WAV.
- Note that the waveform influences the velocity during the motions. Define the waveform so that the specifications of the connected mechanics are adhered to during wave generator output. The velocity is limited by the following factors, among others:
  - Mechanics type
  - Combination of the axes to be moved
  - Current settings for coordinate system and center of rotation
  - Amplitude of the motion
- If the waveform is to be output periodically (i.e., more than one output cycle in succession), the position and velocity at the end of the waveform must be identical to those at the beginning of the waveform.
- If you create a waveform from several segments, the initial position and initial velocity of a segment to be attached must be adapted to the end position and end velocity of the preceding segment.
- Define the waveform so that it can be output with the default output rate. For higher output rates, the C-887 interpolates missing position values, which means that the output waveform may no longer precisely correspond to the definition.

Optional: Deleting the wave table content

1. Make sure that the selected wave table is **not** connected to a wave generator for which the output has been started. Refer to "Configuring a Wave Generator" (p. 111) and "Stopping the Wave Generator Output" (p. 114) for details.
2. Delete the content of the wave tables with the `WCL` command (p. 271).

   The complete content of the selected wave table is deleted. It is **not** possible to delete the wave table content one segment at a time.

INFORMATION

When the C-887 is switched off or rebooted, the wave table content is automatically deleted.

Examples for creating waveforms

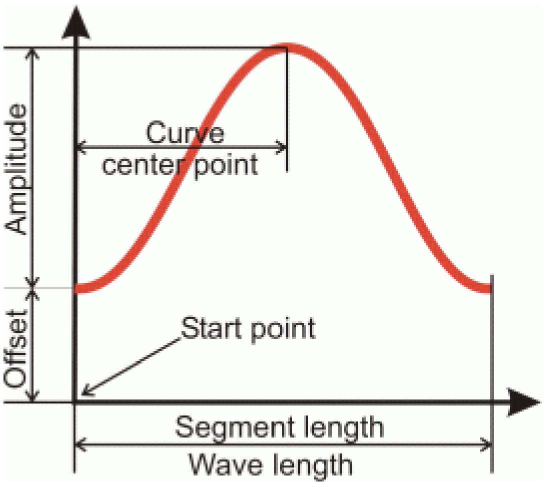
The following examples will help you to create the waveform.

Sine wave 1

- Symmetrical sine wave with offset
- Segment overwrites the contents of the wave table

Command: `WAV 2 X SIN_P 2000 0.2 0.1 2000 0 1000`

<WaveTableID> = 2
<AppendWave> = X
<WaveType> = SIN_P
<SegLength> = 2000
<Amp> = 0.2
<Offset> = 0.1
<WaveLength> = 2000
<StartPoint> = 0
<CurveCenterPoint> = 1000

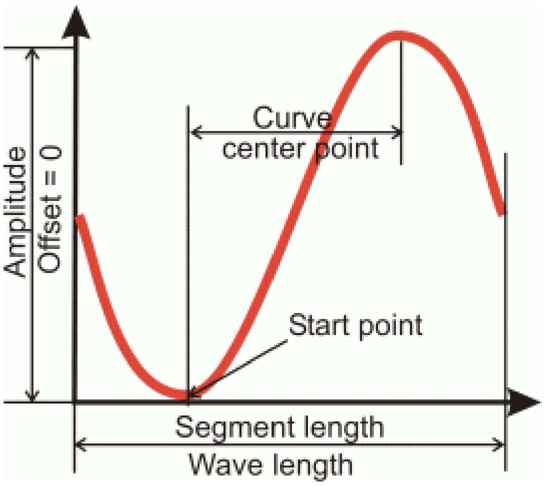


Sine wave 2

- Symmetrical sine wave without offset
- Segment overwrites the contents of the wave table

Command: `WAV 2 X SIN_P 2000 0.3 0 2000 499 1000`

<WaveTableID> = 2
<AppendWave> = X
<WaveType> = SIN_P
<SegLength> = 2000
<Amp> = 0.3
<Offset> = 0
<WaveLength> = 2000
<StartPoint> = 499
<CurveCenterPoint> = 1000

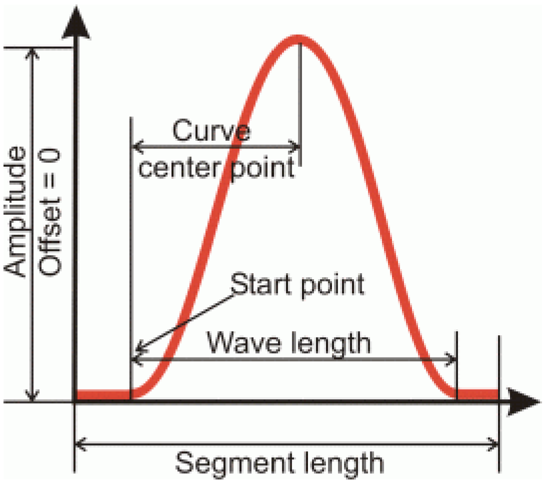


Sine wave 3

- Symmetrical sine wave without offset
- Segment is attached to the content of the wave table

Command: `WAV 2 & SIN_P 2000 0.25 0 1800 100 900`

<WaveTableID> = 2
<AppendWave> = &
<WaveType> = SIN_P
<SegLength> = 2000
<Amp> = 0.25
<Offset> = 0
<WaveLength> = 1800
<StartPoint> = 100
<CurveCenterPoint> = 900

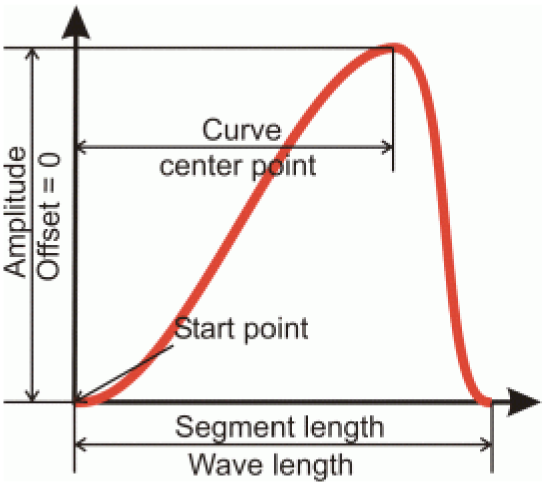


Sine wave 4

- Asymmetrical wave without offset
- Segment overwrites the contents of the wave table

Command: `WAV 3 X SIN_P 4000 0.2 0 4000 0 3100`

<WaveTableID> = 3
<AppendWave> = X
<WaveType> = SIN_P
<SegLength> = 4000
<Amp> = 0.2
<Offset> = 0
<WaveLength> = 4000
<StartPoint> = 0
<CurveCenterPoint> = 3100

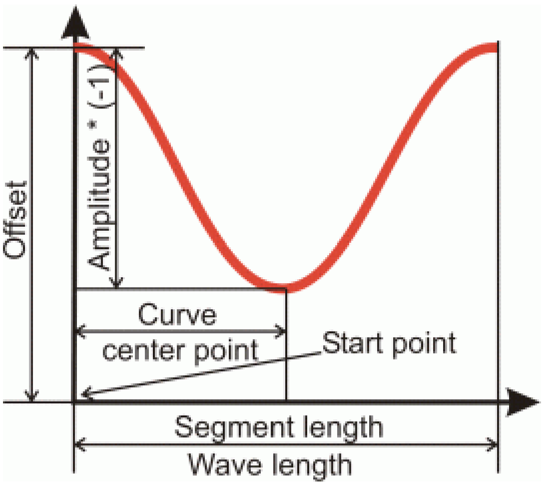


Sine wave 5

- Symmetrical wave with negative amplitude
- Segment overwrites the contents of the wave table

Command: `WAV 1 X SIN_P 1000 -0.3 0.45 1000 0 500`

<WaveTableID> = 1
<AppendWave> = X
<WaveType> = SIN_P
<SegLength> = 1000
<Amp> = –0.3
<Offset> = 0.45
<WaveLength> = 1000
<StartPoint> = 0
<CurveCenterPoint> = 500

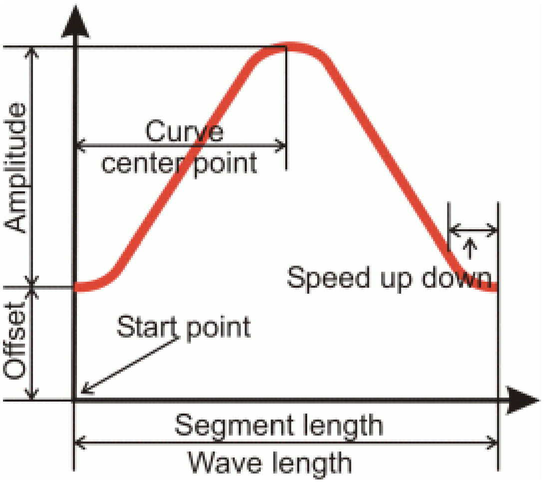


Ramp wave 1

- Symmetrical ramp wave with offset
- Segment overwrites the contents of the wave table

Command: `WAV 4 X RAMP 2000 0.2 0.1 2000 0 300 1000`

<WaveTableID> = 4
<AppendWave> = X
<WaveType> = RAMP
<SegLength> = 2000
<Amp> = 0.2
<Offset> = 0.1
<WaveLength> = 2000
<StartPoint> = 0
<SpeedUpDown> = 300
<CurveCenterPoint> = 1000

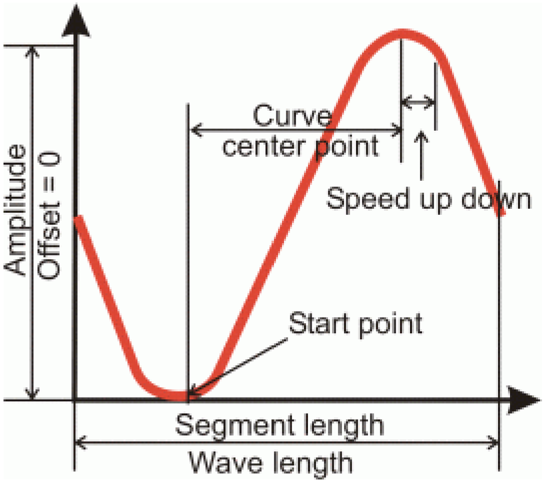


Ramp wave 2

- Symmetrical ramp wave without offset
- Segment overwrites the contents of the wave table

Command: `WAV 4 X RAMP 2000 0.35 0 2000 499 300 1000`

<WaveTableID> = 4
<AppendWave> = X
<WaveType> = RAMP
<SegLength> = 2000
<Amp> = 0.35
<Offset> = 0
<WaveLength> = 2000
<StartPoint> = 499
<SpeedUpDown> = 300
<CurveCenterPoint> = 1000

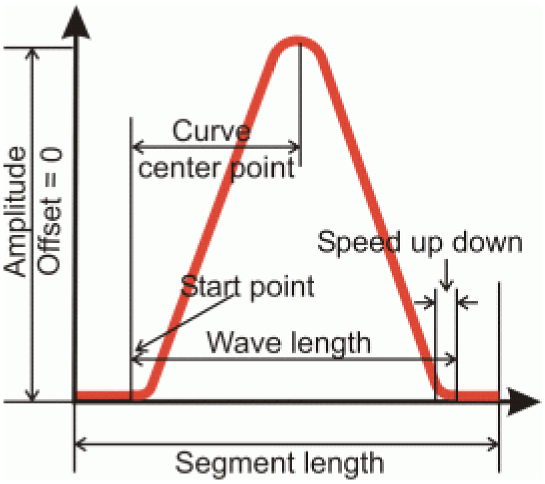


Ramp wave 3

- Symmetrical ramp wave without offset
- Segment overwrites the contents of the wave table

Command: `WAV 5 X RAMP 2000 0.15 0 1800 120 150 900`

<WaveTableID> = 5
<AppendWave> = X
<WaveType> = RAMP
<SegLength> = 2000
<Amp> = 0.15
<Offset> = 0
<WaveLength> = 1800
<StartPoint> = 120
<SpeedUpDown> = 150
<CurveCenterPoint> = 900

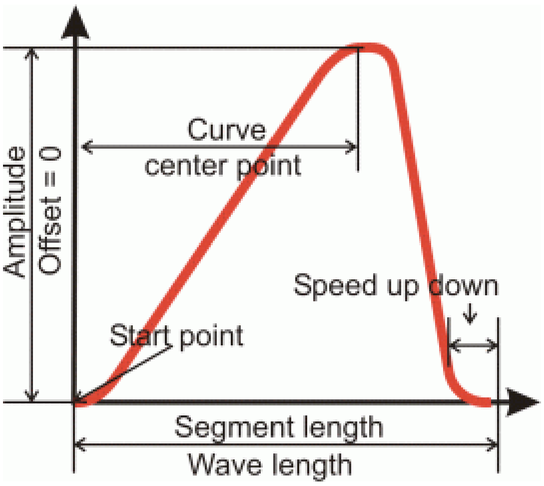


Ramp wave 4

- Asymmetrical ramp wave without offset
- Segment is attached to the content of the wave table

Command: `WAV 5 & RAMP 3000 0.35 0 3000 0 200 2250`

<WaveTableID> = 5
<AppendWave> = &
<WaveType> = RAMP
<SegLength> = 3000
<Amp> = 0.35
<Offset> = 0
<WaveLength> = 3000
<StartPoint> = 0
<SpeedUpDown> = 200
<CurveCenterPoint> = 2250

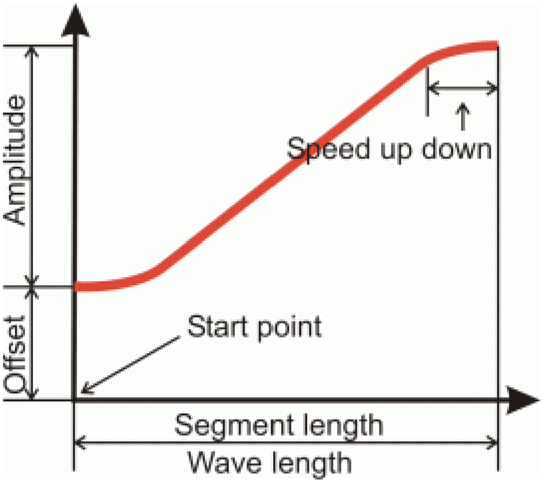


Single scan line 1

- Scan line with offset
- Segment overwrites the contents of the wave table

Command: `WAV 1 X LIN 1500 0.3 0.15 1500 0 370`

<WaveTableID> = 1
<AppendWave> = X
<WaveType> = LIN
<SegLength> = 1500
<Amp> = 0.3
<Offset> = 0.15
<WaveLength> = 1500
<StartPoint> = 0
<SpeedUpDown> = 370

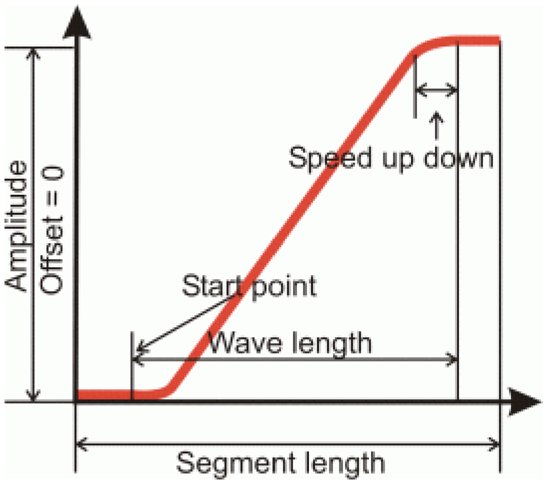


Single scan line 2

- Scan line without offset
- Segment overwrites the contents of the wave table

Command: `WAV 2 X LIN 1500 0.4 0 1100 210 180`

<WaveTableID> = 2
<AppendWave> = X
<WaveType> = LIN
<SegLength> = 1500
<Amp> = 0.4
<Offset> = 0
<WaveLength> = 1100
<StartPoint> = 210
<SpeedUpDown> = 180

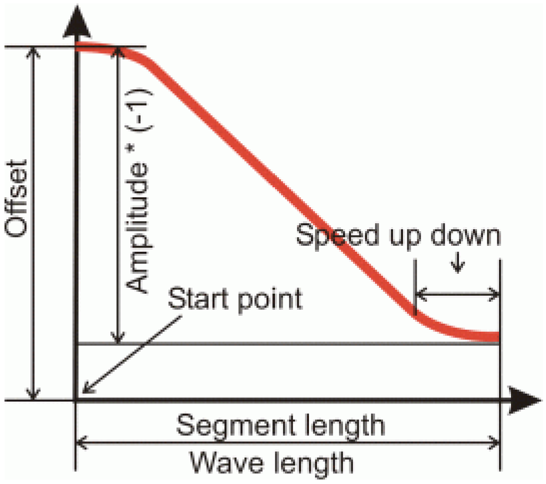


Single scan line 3

- Scan line with negative amplitude
- Segment is attached to the content of the wave table

Command: `WAV 2 & LIN 3000 -0.4 0.5 3000 0 650`

<WaveTableID> = 2
<AppendWave> = &
<WaveType> = LIN
<SegLength> = 3000
<Amp> = –0.4
<Offset> = 0.5
<WaveLength> = 3000
<StartPoint> = 0
<SpeedUpDown> = 650



7.4.4 Configuring a Wave Generator

The wave generator is configured with the following steps:

- Connecting or disconnecting a wave generator and a wave table (p. 111)
- Optional: Setting the number of output cycles (p. 111)
- Optional: Setting the output rate (p. 111)

This manual contains an example for setting the output rate (p. 111).

Connecting or disconnecting a wave generator and a wave table

- Get the current connection of the wave generator and wave table with the `WSL?` command (p. 278).
- Connect or disconnect the wave generator and the wave table:

  a) Make sure that the output has **not** been started for the selected wave generator. Refer to "Stopping the wave generator output" (p. 114) for details.

  b) Use the `WSL` command (p. 278) to connect the selected wave table with the selected wave generator or terminate the connection of the selected generator to a wave table.

  Two or more generators can be connected to the same wave table, but a generator cannot be connected to more than one wave table.

Optional: Setting the number of output cycles

The factory default setting for the number of output cycles is 0. With the default setting, the waveform is output without a time limitation until it is stopped with the `WGO` (p. 272) or `HLT` (p. 193) or `#24` (p. 146) or `STP` (p. 252) commands.

- Send the `WGC?` command (p. 272) to get the current setting for the number of output cycles of the wave generator.
- Set the number of output cycles of the wave generator with the `WGC` command (p. 271).

Optional: Setting the output rate

The individual output cycles of the waveform can be extended by adjusting the output rate. The duration of an output cycle for the waveform can be calculated as follows:

Output duration = servo cycle time * output rate * number of points

where

- Servo Cycle Time for the C-887 is given by the parameter 0x0E000200 (in seconds)
- The output rate is the number of servo cycles that the output of a waveform point lasts; whole multiple of 10 (minimum and default: 10, maximum 1000)

- The number of points is the length of the waveform (i.e., the length of the wave table)

In the case of output rates > 10, the wave points are output with interpolation as standard, in order to avoid position jumps of the axes.

- Send the `WTR?` command (p. 281) to get the current settings for output rate and interpolation.
- Set the output rate and the interpolation with the `WTR` command (p. 278).

Different output rates can be set for the individual wave generators of the C-887.

INFORMATION

- For output rates >10, the C-887 interpolates missing position values linearly. The waveform can then no longer be differentiated at the nodes. Greater jumps of velocity and acceleration can result and induce oscillation in the mechanics. For this reason, use the default output rate (10) if possible.

INFORMATION

If the wave generator identifier 0 is used when setting the output rate with the `WTR` command, the output rate for all wave generators is set to the same value.

Example for setting the output rate

Action	Command	Result
Define a sine curve for wave table 2.	`WAV 2 X SIN_P 2000 0.2 0.1 2000 0 1000`	The length of the waveform and therefore the number of points in the wave table is 2000.
Read out the servo cycle time of the C-887.	`SPA? 1 0x0E000200`	The servo cycle time of the C-887 is 100 µs
Read the current output rate.	`WTR?`	Default value for the output rate = 10 (each point in the wave table is output during 10 servo cycles) Duration of an output cycle (see calculation formula above): 0.0001 s • 10 • 2000 = 2 s

Action	Command	Result
Triple the number of servo cycles per point in the wave table for all wave generators.	`WTR 0 30 1`	Duration of an output cycle (see calculation formula above): 0.0001 s • 30 • 2000 = 6 s Between the output of the points, the C-887 interpolates linearly.

7.4.5 Starting and Stopping Output

The wave generator outputs absolute target positions.

- Starting the wave generator output (p. 114)
- Stopping the wave generator output (p. 114)
- Optional: Getting the activation state and status of the wave generator (p. 114)
- Optional: Starting data recording during the wave generator output (p. 115)

This manual contains examples for starting/stopping the wave generator output (p. 115).

INFORMATION

When the wave generator output is active, commands for starting or configuring motion and executing corresponding macros are **not** permitted.

INFORMATION

During the wave generator output, the C-887 continuously checks whether the motion is still possible. In the following cases, the C-887 stops the motion abruptly and sets an error code:

- The target positions to be output cannot be achieved.
- The necessary velocity cannot be achieved.
- The motion would cause a collision.

➢ Use the `WGS?` command to get the current status of the wave generators, especially the index of the waveform points at which an error has occurred.

➢ Get the error code of the last error that occurred with the `ERR?` command (p. 164).

Requirements

✓ You have created the desired waveform (p. 103).

✓ You have connected the wave generator with the corresponding wave table (p. 111).

Starting the wave generator

- Start the wave generator output with the `WGO` command (p. 272).

  All wave generators whose output has to be active at the same time must be started in the same command. For the axes of the motion platform of the hexapod whose wave generator is not started, the last valid target position is always commanded.

  The output takes place synchronously with the servo cycles of the C-887.

  When the wave generator output is started, a data recording cycle automatically starts.

Stopping the wave generator output

- Stop the wave generator output by sending one of the following commands:
  - `WGO` (p. 272) stops the specified wave generators
  - `STP` (p. 252) stops all wave generators
  - `#24` (p. 146) stops all wave generators
  - `HLT` (p. 193) stops the wave generators of the specified axes

  When the wave generator output is stopped by sending `STP`, `#24`, or `HLT`, the C-887 sets the error code 10 (get with the `ERR?` command (p. 164)).

When the number of output cycles has been limited (p. 111), the wave generator output is automatically stopped when the specified number of cycles is reached.

INFORMATION

Exiting the PC software does **not** stop the wave generator output.

Optional: Getting the activation state of the wave generator

- Get whether the wave generator output is running with the `#9` command (p. 145).
- Get the last-commanded start/stop settings of the wave generator with the `WGO?` command (p. 273).

  The response to `WGO?` is also affected by stopping the wave generator output with `#24` (p. 146), `STP` (p. 252), or `HLT` (p. 193) and the end of a specified number of output cycles.
- Get the status of the wave generator with the `WGS?` command:
  - Is the wave generator running?
  - How many cycles has the wave generator output since the last start?
  - Error code of the last error that occurred during the output
  - Index of the waveform point at which the error occurred

Optional: Starting data recording during the wave generator output

- Start the data recording during the wave generator output by sending the command `WGR` (p. 274).

When the wave generator output is started (p. 114), an initial data recording cycle automatically starts.

The recorded data can be read out with the `DRR?` command (p. 160). For further information, see "Data Recorder" (p. 97).

Examples for starting/stopping the wave generator output

Action	Command	Result
Define a sine curve for wave table 4.	`WAV 4 X SIN_P 2000 1 0 2000 0 1000`	The length of the waveform and thus the number of points in the wave table is 2000.
Define a sine curve for wave table 5.	`WAV 5 X SIN_P 2000 0.2 0 2000 0 1000`	The length of the waveform and thus the number of points in the wave table is 2000.
Connect wave generator 1 (axis X) with wave table 4. Connect wave generator 3 (axis Z) with wave table 5.	`WSL 1 4 3 5`	Prerequisite for wave generator output fulfilled: No wave generator output is possible without allocation of a wave table.
Start wave generators 1 and 3.	`WGO 1 1 3 1`	The waveforms defined in wave tables 4 and 5 are output. The last valid target positions are commanded for the Y, U, V, and W axes of the motion platform of the hexapod, whose wave generators have not been started.
Stop wave generators 1 and 3.	`WGO 1 0 3 0`	The output of the wave points (and therefore motion of the hexapod) is stopped.

All commands except for the last two lines as in the example above; the last two lines are replaced by the following commands:

Action	Command	Result
Limit the number of output cycles for wave generators 1 and 3 to 100.	`WGC 1 100 3 100`	A command to stop the wave generators is not necessary.
Start wave generators 1 and 3.	`WGO 1 1 3 1`	The output of the wave generators ends automatically after 100 output cycles.

7.4.6 Application Tips: Loading Customer-Specific Data

You can load customer-specific data (time series) from files from the PC into the C-887 and use it as user-defined waves. Before loading, you have to convert the data into GCS array format. To load the converted data to the C-887, use the PI Wave Generator Tool or the PI Frequency Generator Tool (both available in PIMikroMove). The use of the PI Wave Generator Tool is described in the following; for the use of the PI Frequency Generator Tool, see the PIMikroMove manual (SM148E).

Converting data to the GCS array format

The GCS array format is based on ASCII characters. A GCS array file consists of a header section and a data table section. A comma or a period can be used as a decimal sign. For a detailed description of the GCS array format, see the separate manual (SM146E).

Convert the data of your time series to GCS array format as follows:

1. Create a text file with the file extension ".dat" on the PC, e.g., "New_GCS_Array.dat".
2. Add the header to the file. The structure and contents of the header must be as follows:

   Template for the GCS array header:

   `[GCS_ARRAY `**Target Positions XY**`]`

   `# VERSION = 1`

   `# TYPE = 1`

   `# SEPARATOR = `**32**

   `# DIM = 2`

   `# SAMPLE_TIME = `**0.001**

   `# NDATA = `**256**

   `#`

   `# NAME0 = `**TARGET POSITION X**

   `# NAME1 = `**TARGET POSITION Y**

   `#`

   `# DISP_UNIT0 = `**mm**

   `# DISP_UNIT1 = `**mm**

   `# END_HEADER`

   Adapt the following information in the header to your data (**marked** in the list above):

   `DIM`
   Number of columns in the data table (i.e., number of time series), the minimum is 1

   `NDATA`
   Number of lines in the data table (i.e., number of data points per time series)

`NAME%`
Optional: Description of the column % (whereby % is a positive integer value < DIM; the counting begins at 0)

`DISP_UNIT%`
Optional: Description of the physical unit for column % (whereby % is a positive integer value < DIM; the counting begins at 0)

`SAMPLE_TIME`
Sample time of the data (in s), i.e., the time interval between two data points in the time series

`SEPARATOR`
Separator for the columns in the data table. Must be given as a decimal ASCII character (e.g., 9 for TAB or 32 for SPACE)

`[GCS_ARRAY %]`
Optional: % is the name of the data table

3. Add your data to the file according to the header information for `DIM`, `NDATA`, and `SEPARATOR`:

   Data table section of the GCS array file, spaces are shown here as SP:

   ```
   Value1_TimeSeries0SPValue1_TimeSeries1SP…
   Value2_TimeSeries0SPValue2_TimeSeries1SP…
   Value3_TimeSeries0SPValue3_TimeSeries1SP…
   …
   ```

   The following examples shows a complete GCS array with the data for two time series. The first time series, "TARGET POSITION X", contains the values 1/2/3, and the second time series, "TARGET POSITION Y", contains the values 0.1/0.2/0.3.

   ```
   # REM data recorded with C-887 controller
   #
   # VERSION = 1
   # TYPE = 1
   # SEPARATOR = 32
   # DIM = 2
   # SAMPLE_TIME = 0.009009
   # NDATA = 3
   #
   # NAME0 = TARGET POSITION X
   # NAME1 = TARGET POSITION Y
   #
   # DISP_UNIT0 = mm
   ```

```
# DISP_UNIT1 = mm
# END_HEADER
1 0.1
2 0.2
3 0.3
```

The result of the conversion is a GCS array file with the file extension ".dat" that contains the time series data. The file can be imported to the C-887 with the PI Wave Generator Tool (see below) or the PI Frequency Generator Tool.

Loading GSC array data

The PI Wave Generator Tool is available in PIMikroMove. For further information on the PI Wave Generator Tool, see the PIMikroMove manual (SM148E).

1. Open the PI Wave Generator Tool from the main window of PIMikroMove via the ***C-887 > Show wave generator...*** menu item.
2. In the main window of the PI Wave Generator Tool, click ***Load Data Set*** to open the ***Load Data Set for Wave Tables*** window.

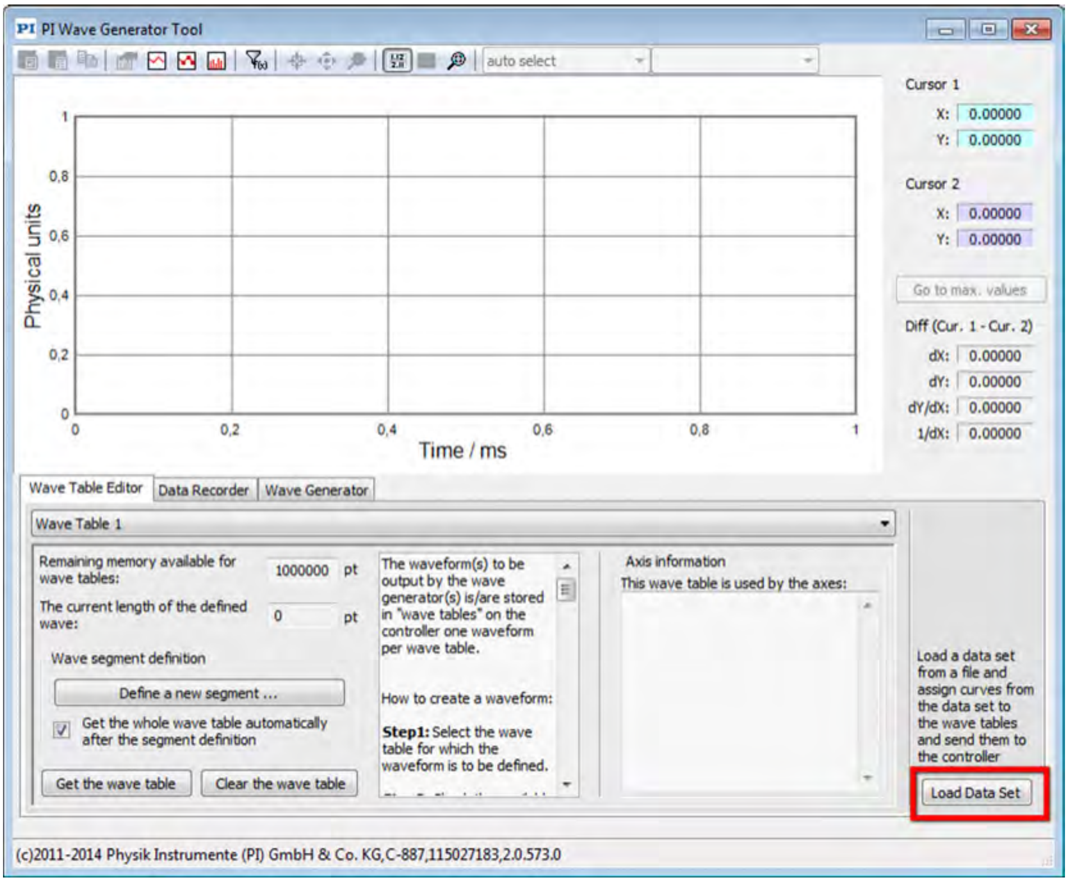


Figure 12: Main window of the PI Wave Generator Tool

3. Import the GCS array file in the ***Load Data Set for Wave Tables*** window.

   The time series data that has been loaded from the file is shown in the graphic field. The figure below shows the data from the example in "Converting data to the GCS array format".

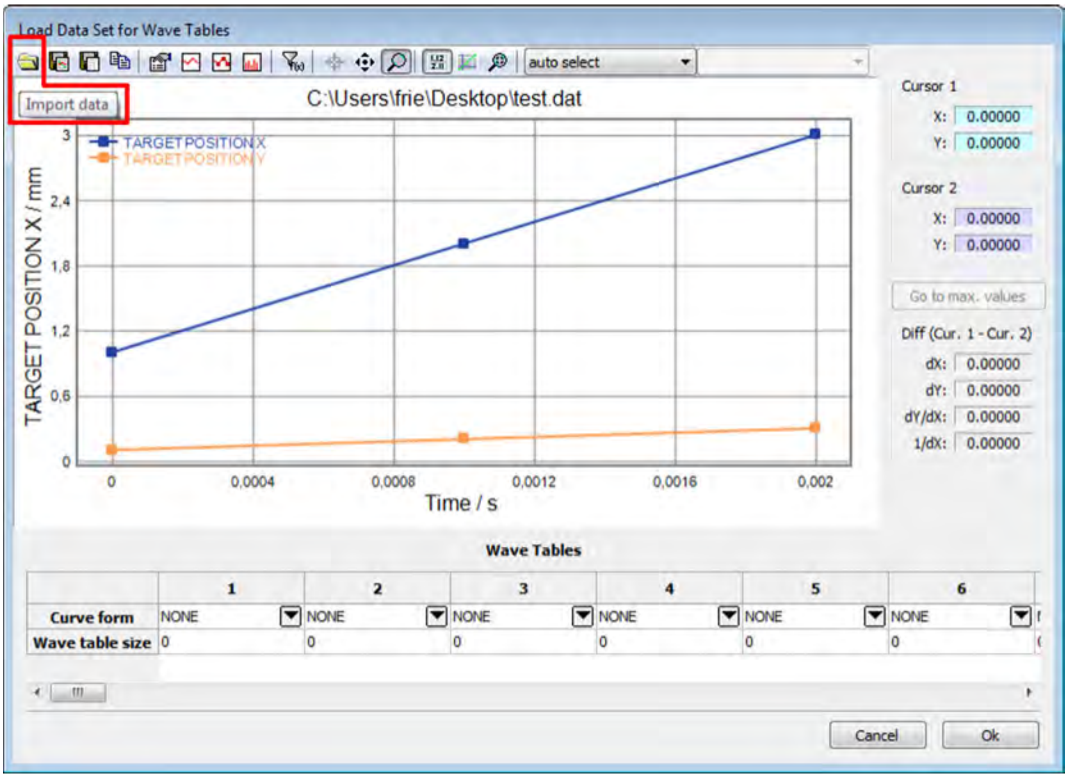


Figure 13: "Load Data Set for Wave Tables" window with graphic display of the loaded data

4. Assign the individual time series to the wave tables of the C-887. For the assignment, open the selection menu of a wave table and select the desired data (see figure below).

   Note that the data is not sent to the C-887 until you click ***Ok*** in the lower right corner of the window.

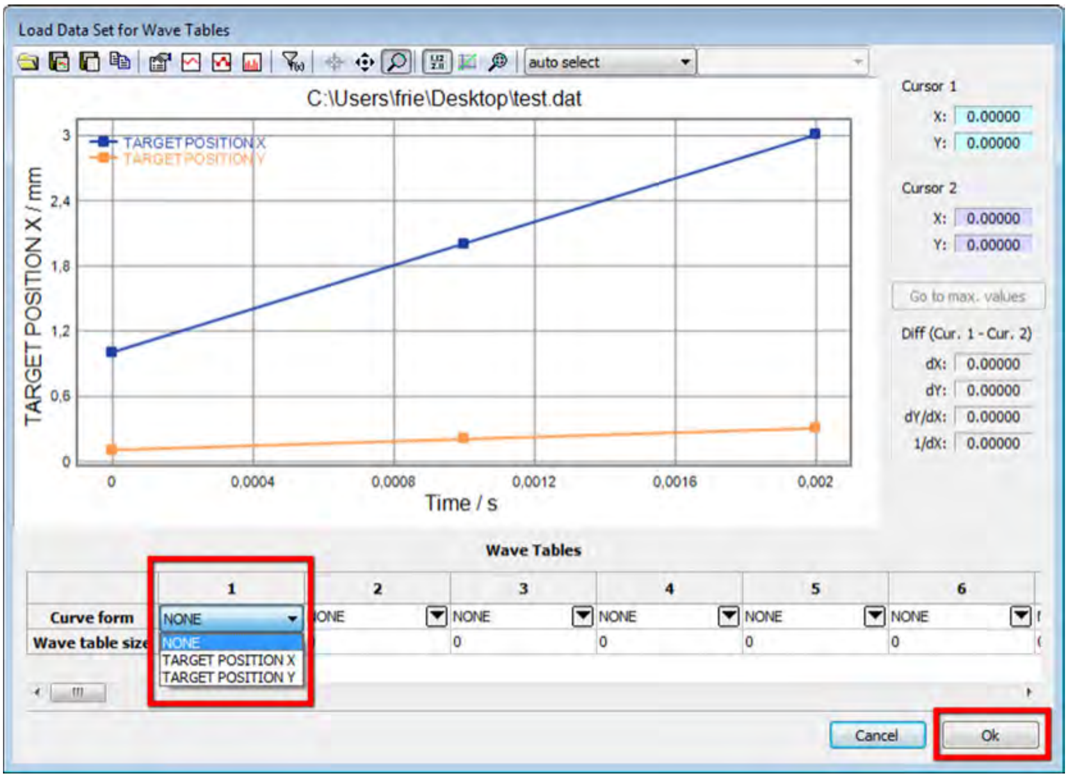


Figure 14: Assignment of the loaded data to wave tables, here for wave table 1

7.4.7 Application Tips: Using Macros for the Wave Generator

The following settings for the wave generator can only be changed in the volatile memory of the C-887 and are lost when the C-887 is switched off or rebooted:

- Contents of the wave tables (defined with WAV)
- Connection of wave tables with wave generators (set with WSL)
- Output rate (set with WTR)
- Number of output cycles (set with WGC)

You can permanently save the settings of the wave generator in the C-887 with the macro functionality of the C-887. You can also use a startup macro to configure the wave generator and start the output each time that the C-887 is switched on or rebooted.

For more information on macro functionality, see "Controller Macros" (p. 122).

In the following example, two macros are used to configure and start the wave generator output for the axes U and V (wave generators 4 and 5) of the hexapod:

- The "UVdata" macro contains the definition of the waveforms for wave tables 1 and 2 (WAV commands).
- The "Start" macro executes the following actions:

  a) Write the waveforms in wave tables 1 and 2 by calling up the "UVdata" macro.

b) Connect wave table 1 with wave generator 4 and wave table 2 with wave generator 5 (WSL command).
c) Set the output rate for wave generators 4 and 5 to 10, with linear interpolation (WTR command; the setting for the interpolation must also be set but is ignored when it is output with an output rate of 10).
d) Set the number of output cycles for wave generators 4 and 5 to 100 (WGC command).
e) Optional but recommended for preventing axes from jumping: Command axes U and V to the starting position of the wave generator output. Then wait until the target positions have been reached.
f) Start the output for wave generators 4 and 5 - and therefore the motion of axes U and V (WGO command).

Contents of the "UVdata" macro:

```
WAV 1 X PNT 1 1 0
WAV 1 & PNT 1 1 5.654625319357E-06
WAV 1 & PNT 1 1 3.091004951757E-05
[…]
WAV 1 & PNT 1 1 0.000148233661

WAV 2 X PNT 1 1 0
WAV 2 & PNT 1 1 -4.742444189387E-06
WAV 2 & PNT 1 1 -3.027352414704E-05
[…]
WAV 2 & PNT 1 1 -0.000257643502
```

Content of the "Start" macro, whereby `startposU` and `startposV` designate the starting position of the wave generator output:

```
MAC START UVdata
WSL 4 1 5 2
WTR 4 10 1 5 10 1
WGC 4 100 5 100
MOV U startposU V startposV
WAC ONT? U = 1
WAC ONT? V = 1
WGO 4 1 5 1
```

Remarks:

In the example macro "UVdata", only one wave point (= target position) is given per WAV command for a better overview. For faster processing, it is recommended to define more than one point per WAV command (max. 256 characters are permitted per command line).

Waveform definitions can be easily created by copying the contents of the log window of PIMikroMove to the macro:

1. Open the log window via the ***C-887 > Log window...*** menu item in the main window of PIMikroMove.
2. Open the PI Frequency Generator Tool or the PI Wave Generator Tool from the ***C-887*** menu in the main window of PIMikroMove.
3. Define a waveform with the tool and send the waveform to the C-887.

   The sent commands are listed in the log window.

   For example, you can define a sine wave with a particular frequency using the PI Frequency Generator Tool. You can then copy the corresponding WAV ... PNT commands from the log window to a macro. The WAV … PNT commands in the log window are created by the PI_WAV_PNT() function of the PI GCS2 DLL used by PIMikroMove. The PI_WAV_PNT() function automatically distributes the wave points to the single WAV … PNT commands.

When the example macro "Start" is used as a startup macro, a reference move can be necessary before the wave generator output is started. In this case, the following lines must be added to the macro before the `MOV U startposU V startposV` line:

```
FRF X
WAC FRF? X = 1
```

The lines start a reference move and wait until the reference move has been successfully completed.

7.5 Controller Macros

7.5.1 Overview: Macro Functionality and Example Macros

The C-887 can save and process command sequences as macros.

The following functionalities make macros an important tool in many application areas:

- Several macros can be saved at the same time.
- Any macro can be defined as the startup macro. The startup macro runs each time the C-887 is switched on or rebooted.
- Processing and stopping a macro can be linked to conditions. This makes loops possible.
- Macros can call up themselves or other macros.
- Variables (p. 131) can be set for the macro and in the macro itself and used in different operations.
- Input signals can be evaluated for conditions and variables.

You will find example macros in this manual for the following tasks:

- Moving an axis back and forth (p. 126)
- Moving an axis with a variable travel range back and forth (p. 128)
- Triggering a reference move for the hexapod with a startup macro (p. 130)

7.5.2 Commands and Parameters for Macros

Commands

The following commands are specially available for handling macros or for use in macros:

Command	Syntax	Function
ADD (p. 150)	ADD <Variable> <FLOAT1> <FLOAT2>	Can only be used in macros. Adds two values and saves the result to a variable (p. 131).
CPY (p. 152)	CPY <Variable> <CMD?>	Can only be used in macros. Copies a command response to a variable (p. 131).
DEL (p. 154)	DEL <uint>	Can only be used in macros. Causes a delay of <uint> milliseconds.
JRC (p. 199)	JRC <Jump> <CMD?> <OP> <Value>	Can only be used in macros. Triggers a relative jump of the macro execution pointer depending on a condition.
MAC (p. 223)	MAC BEG <macroname>	Starts the recording of a macro with the name *macroname* on the controller. *macroname* can consist of up to 8 characters.
	MAC DEF <macroname>	Defines the specified macro as the startup macro.
	MAC DEF?	Gets the startup macro.
	MAC DEL <macroname>	Deletes the specified macro.
	MAC END	Ends the macro recording.
	MAC ERR?	Reports the last error that occurred while the macro was running.
	MAC FREE?	Gets the free memory space for macro recording.
	MAC NSTART <macroname> <uint> [{<String>}]	Starts the specified macro n times in succession (n = number of runs). The values of local variables can be set for the macro with <String>.
	MAC START <macroname> [{<String>}]	Runs the specified macro. The values of local variables can be set for the macro with <String>.
MAC? (p. 226)	MAC? [<macroname>]	Lists all macros or the content of a specified macro.
MEX (p. 227)	MEX <CMD?> <OP> <Value>	Can only be used in macros. Stops the macro depending on a condition.

Command	Syntax	Function
RMC? (p. 239)	RMC?	Lists macros which are currently running.
STP (p. 252)	STP	Stops the macro.
VAR (p. 259)	VAR <Variable> <String>	Sets a variable (p. 131) to a certain value or deletes it.
VAR? (p. 260)	VAR? [{<Variable>}]	Gets variable values.
WAC (p. 264)	WAC <CMD?> <OP> <Value>	Can only be used in macros. Waits until a condition is met.
#8 (p. 145)	-	Tests if a macro is running on the controller.
#24 (p. 146)	-	Stops the macro.

INFORMATION

A maximum of 256 characters are permitted per command line.

Parameters

The following parameter is available for working with macros:

Parameters	Description and Possible Values
Ignore Macro Error? 0x72	Determines whether the controller macro is stopped if an error occurs when it is running. ▪ 0 = Stop macro when error occurs (default) ▪ 1 = Ignore error

7.5.3 Working with Macros

Work with macros comprises the following:

- Recording macros (p. 125)
- Starting macros (p. 127)
- Stopping macros (p. 129)
- Configuring a startup macro (p. 130)
- Deleting macros (p. 130)

INFORMATION

It is recommended to use the ***Controller macros*** tab in PIMikroMove when working with controller macros. There you can record, start, and manage controller macros easily. Refer to the PIMikroMove manual for details.

Recording a macro

INFORMATION

The C-887 can save an unlimited number of macros at the same time. A maximum of 10 nesting levels are possible in macros.

INFORMATION

Basically all GCS commands (p. 133) can be included in a macro. Exceptions:

- `RBT` for rebooting the C-887
- `MAC BEG` and `MAC END` for macro recording
- `MAC DEL` for deleting a macro
- `#27` for stopping the C-887

Query commands can be used in macros in conjunction with the `CPY`, `JRC`, `MEX`, and `WAC` commands. Otherwise they have no effect, since macros do not transmit any responses to interfaces.

INFORMATION

To make the use of macros more flexible, you can use local and global variables in macros. For further information, see "Variables" (p. 131).

INFORMATION

The number of write cycles in the nonvolatile memory is restricted by the limited lifetime of the memory chip.

- ➢ Only record macros if it is necessary.
- ➢ Use variables (p. 131) in macros to make macros more flexible, and give the corresponding variable values when starting macro execution.
- ➢ Contact our customer service department (p. 343) if the C-887 shows unexpected behavior.

1. Start the macro recording.
   - If you are working with PITerminal or in the ***Command entry*** window of PIMikroMove: Send the `MAC BEG macroname` command, where *macro name* indicates the name of the macro.
   - If you are working in PIMikroMove in the ***Controller macros*** tab: Click the ***Create new empty macro*** icon to create a tab for entering a new macro. Do **not** enter the `MAC BEG macroname` command.
2. Enter the commands to be included in the *macro name* macro line by line, using the normal command syntax.

   Macros can call themselves or other macros at several nesting levels.
3. End the macro recording.
   - If you are working with PITerminal or in the ***Command entry*** window of PIMikroMove: Send the `MAC END` command.
   - If you are working in PIMikroMove in the ***Controller macros*** tab: Do **not** enter the `MAC END` command. Click the ***Send macro to controller*** icon and enter the name of the macro in a separate dialog window.

   The macro was stored in the nonvolatile memory of the C-887.
4. If you want to check whether the macro was recorded correctly:

   If you are working with PITerminal or in the ***Command entry*** window of PIMikroMove:
   - Get which macros are saved in the C-887 by sending the `MAC?` command.
   - Get the contents of the *macro name* macro by sending the `MAC? macroname` command.

   If you are working in PIMikroMove in the ***Controller macros*** tab:
   - Click the ***Read list of macros from controller*** icon.
   - Mark the macro to be checked in the list on the left-hand side and click the ***Load selected macro from controller*** icon.

Example: Moving an axis back and forth

INFORMATION

When macros are recorded on the ***Controller macros*** tab in PIMikroMove, the `MAC BEG` and `MAC END` commands must be left out.

The axis X is to move back and forth. For this purpose, 3 macros are recorded. Macro 1 starts motion in a positive direction and waits until the axis has reached the target position. Macro 2 does this task for the negative direction of motion. Macro 3 calls up macro 1 and 2.

➢ Record the macros by sending:

```
MAC BEG macro1
```

```
MVR X 12.5
WAC ONT? X = 1
MAC END
MAC BEG macro2
MVR X -12.5
WAC ONT? X = 1
MAC END
MAC BEG macro3
MAC START macro1
MAC START macro2
MAC END
```

Starting a macro

INFORMATION

When a macro is running on the C-887, commands can be sent to the C-887 via all communication interfaces. Commands are processed in the order that they arrive. The processing time for the individual commands depends on the command. The execution of a time-consuming command that was sent via a communication interface can therefore lead to a pause in macro execution. Macros are therefore not suitable for time-critical running of defined dynamics profiles, for example.

- Do not use a macro for running a fixed, time-critical dynamics profile but instead, the wave generator (p. 99) or the consecutive MOV commands that are stored in a buffer (p. 36).
- Avoid using a control unit (C-887.MC2 or C-887.MC) when a macro is running.
- If possible, do not start or use PC software such as PIMikroMove while running the macro or driver for use with NI LabVIEW software.
- If possible, avoid sending commands when a macro is running.

INFORMATION

It is not possible to run several macros simultaneously. Only one macro can be run at a time.

INFORMATION

You can link the macro execution to conditions with the `JRC` and `WAC` commands. The commands must be included in the macro.

In the following, the PITerminal or the ***Command entry*** window of PIMikroMove is used to enter commands. Details on working with the ***Controller macros*** tab in PIMikroMove are found in the PIMikroMove manual.

1. If the macro is to continue running despite an error:
   - Set the ***Ignore Macro Error?*** parameter (ID 0x72) correspondingly: Send the `SPA 1 0x72 Status` command, whereby *Status* can have the value 0 or 1 (0 = Stop macro when error occurs (default); 1 = Ignore macro error).

   Further information on changing parameters can be found in "Adapting Settings" (p. 299).
2. Start the macro:
   - If the macro is to be run once, send the `MAC START macroname string` command, whereby *macroname* indicates the name of the macro.
   - If the macro is to be run n times, send the `MAC NSTART macroname n string` command, whereby *macroname* indicates the name of the macro and *n* indicates the number of runs.

   string stands for the values of local variables. The values only have to be specified when the macro contains corresponding local variables. The sequence of the values in the input must correspond to the numbering of the appropriate local variables, starting with the value of local variable 1. The individual values must be separated from each other by spaces.
3. If you want to check whether the macro is running:
   - Query whether a macro is running on the controller by sending the `#8` command.
   - Query the name of the macro that is currently running on the controller by sending the `RMC?` command.

Example: Moving an axis with a variable travel distance back and forth

INFORMATION

When macros are recorded on the ***Controller macros*** tab in PIMikroMove, the `MAC BEG` and `MAC END` commands must be left out.

The axis X is to move back and forth. The travel to the left and to the right is to be variably adjustable without having to change the used macros. Local and global variables are therefore used.

1. Create the global variables LEFT and RIGHT by sending:

   `VAR LEFT 5`

   `VAR RIGHT 15`

   LEFT therefore has the value 5, and RIGHT has the value 15. These values can be changed at any time, e.g., by sending the `VAR` command again.

– Create the global variables again each time the C-887 is switched on or rebooted, since they are only written to the volatile memory of the C-887.

2. Record the MOVLR macro by sending:

 `MAC BEG movlr`

 `MAC START movwai ${LEFT}`

 `MAC START movwai ${RIGHT}`

 `MAC END`

 MOVLR successively starts the MOVWAI macro (which is still to be recorded) for both directions of motion. The values of the global variables LEFT and RIGHT are used when MOVWAI is started, to set the value of the local variable 1 contained in MOVWAI (dollar signs and braces are necessary for the local variable 1 in the macro to actually be replaced by the *value* of the global variable and not by its *name*).

3. Record the MOVWAI macro by sending:

 `MAC BEG movwai`

 `MOV X $1`

 `WAC ONT? X = 1`

 `MAC END`

 MOVWAI moves axis X to the target position which is specified by the value of the local variable 1 and waits until the axis has reached the target position.

4. Run the MOVLR macro by sending:

 `MAC NSTART movlr 5`

 The MOVLR macro is executed five times in succession, i.e., axis X alternately moves to the positions 5 and 15 five times. You can also select any other value for the number of executions.

Stopping a macro

INFORMATION

You can link the stopping of the macro execution to a condition with the `MEX` command. The command must be included in the macro.

In the following, the PITerminal or the ***Command entry*** window of PIMikroMove is used to enter commands. Details on working with the ***Controller macros*** tab in PIMikroMove are found in the PIMikroMove manual.

- Stop the macro execution with the `#24` or `STP` commands.
- If you want to check whether an error has occurred during the macro execution, send the `MAC ERR?` command. The response shows the last error that has occurred.

Configuring a startup macro

Any macro can be defined as the startup macro. The startup macro is executed each time the C-887 is switched on or rebooted.

INFORMATION

Deleting a macro does not delete its selection as a startup macro.

In the following, the PITerminal or the ***Command entry*** window of PIMikroMove is used to enter commands. Details on working with the ***Controller macros*** tab in PIMikroMove are found in the PIMikroMove manual.

- Define a macro as the startup macro with the `MAC DEF macroname` command, whereby *macroname* indicates the name of the macro.
- If you want to cancel the selection of the startup macro and do not want to define another macro as the startup macro, only send `MAC DEF`.
- Get the name of the currently defined startup macro by sending the `MAC DEF?` command.

Example: Triggering a reference move for the hexapod with a startup macro

INFORMATION

When macros are recorded on the ***Controller macros*** tab in PIMikroMove, the `MAC BEG` and `MAC END` commands must be left out.

The STARTCL macro starts a reference move. STARTCL is defined as a startup macro so that the hexapod executes the reference move immediately after switching on.

- Send:

```
MAC BEG startcl
DEL 1000
FRF X
MAC END
MAC DEF startcl
```

Deleting a macro

INFORMATION

A macro cannot be deleted while it is running.

In the following, the PITerminal or the ***Command entry*** window of PIMikroMove is used to enter commands. Details on working with the ***Controller macros*** tab in PIMikroMove are found in the PIMikroMove manual.

➢ Delete a macro with the `MAC DEL macroname` command, whereby *macro name* indicates the name of the macro.

7.5.4 Variables

For more flexibility in programming, the C-887 supports the use of variables in macros. While global variables can be used for all macros, local variables are only valid for a particular macro. The number of global or local variables is not restricted.

Variables are present in volatile memory (RAM) only. The variable values are of the STRING data type.

Working with variables comprises the following:

- Generating variables:
  - Local variables: see "Specifics of Local Variables"
  - Global variables: within a macro with the ADD (p. 150), CPY (p. 152) and VAR (p. 259) commands, outside a macro with VAR
- Changing variable values: within a macro with the ADD, CPY and VAR commands
- Getting variable values: with VAR? within or outside a macro
- Deleting variables: with VAR within a macro; global variables even outside a macro

The following conventions apply to variable names:

- Variable names may not contain special characters (especially not “$”).
- The maximum number of characters is 8.
- Names of global variables can consist of characters A to Z and 0 to 9. They must start with an alphabetic character.
- Names of local variables may not contain alphabetic characters. Possible characters are 0 to 9.
- The variable name can also be specified via the value of another variable.

If the value of a variable is to be used, the notation must be as follows:

- The variable name must be preceded by the dollar sign ($).
- Variable names consisting of multiple characters must be put in braces.

If the variable name consists of a single character, the braces can be left out.

Note that if the braces are left out of variable names consisting of multiple characters, the first character after the “$” is interpreted as the variable name.

Specifics of Local Variables

- The names of local variables used in a macro must form a continuous series. Example of permissible designation: 1, 2, 3, 4. Designation with, e.g., 1, 2, 5, 6 is not allowed.
- The values of local variables are given as arguments <String> of the `MAC START` or `MAC NSTART` command when starting the macro.

  The command formats are:

  `MAC START <macroname> [{<String>}]`

  `MAC NSTART <macroname> <uint> [{<String>}]`

  <String> stands for the value of a local variable contained in the macro. The sequence of the values in the input must correspond to the numbering of the appropriate local variables, starting with the value of the local variable 1. The individual values must be separated from each other by spaces. A maximum of 256 characters are permitted per command line. <String> can be given directly or via the value of another variable.

  <uint> gives the number of times the macro is to be run. See the `MAC` command (p. 223) description for more information.
- The local variable 0 is read-only. Its value indicates the number of arguments (i.e., values of local variables) set when starting the macro.
- As long as the macro is running, the values of local variables can be queried with `VAR?`.

8 GCS Commands

In this Chapter

8.1 Notation

The following notation is used to define the GCS syntax and to describe the commands:

<...>	Angle brackets indicate an argument of a command, can be an element identifier or a command-specific parameter
[…]	Square brackets indicate an optional entry
{…}	Braces indicate a repetition of entries, i.e., that it is possible to access more than one element (e.g., several axes) in one command line.
LF	LineFeed (ASCII char #10), is the default termination character (character at the end of a command line)
SP	Space (ASCII char #32), indicates a space character
"..."	Quotation marks indicate that the characters enclosed are returned or to be entered.

8.2 GCS Syntax for Syntax Version 2.0

A GCS command consists of 3 characters, e.g., CMD. The corresponding query command has a question mark at the end, e.g., CMD?

Command mnemonic:

CMD ::= character1 character2 character3 [?]

Exceptions:

- Single-character commands, e.g., fast query commands, consist only of one ASCII character. The ASCII character is written as combination of # and the character code in decimal format, e.g., as `#24`.
- `*IDN?` (for GPIB compatibility).

The command mnemonic is not case-sensitive. The command mnemonic and all arguments (e.g., axis identifiers, channel identifiers, parameters, etc.) must be separated from each other by a space (SP). The command line ends with the termination character (LF).

CMD[{{SP}<Argument>}]LF

CMD?[{{SP}<Argument>}]LF

Exception:

- Single-character commands are not followed by a termination character. However, the response to a single-character command is followed by a termination character.

The argument <AxisID> is used for the logical axes of the controller. Depending on the controller, an axis identifier can consist of up to 16 characters. All alphanumeric characters and the underscore are allowed. Refer to "Commandable Elements" (p. 25) for the identifiers supported by the C-887.

Example 1:

Axis 1 is to be moved to position 10.0. The unit depends on the controller (e.g., µm or mm).

Send: `MOV`SP`1`SP`10.0`LF

More than one command mnemonic per line is not allowed. Several groups of arguments following a command mnemonic are allowed.

Example 2:

Two axes connected to the same controller are to be moved:

Send: `MOV`SP`1`SP`17.3`SP`2`SP`2.05`LF

When a part of a command line cannot be executed, the line is not executed at all.

When all arguments are optional and are not specified, the command is executed for all possible argument values.

Example 3:

All parameters in the volatile memory are to be reset.

Send: `RPA`LF

Example 4:

The position of all axes is to be queried.

Send: `POS?`LF

The response syntax is as follows:

[<Argument>[{SP<Argument>}]"="]<Value>LF

With multi-line replies, the space preceding the termination character is left out of the last line:

{[<Argument>[{SP<Argument>}]"="]<Value>SPLF}

[<Argument>[{SP<Argument>}]"="]<Value>LF for the last line!

The arguments are listed in the response in the same order as in the query command.

Query command:

CMD?SP<Arg3>SP<Arg1>SP<Arg2>LF

Response to this command:

<Arg3>"="<Val3>SPLF

<Arg1>"="<Val1>SPLF

<Arg2>"="<Val2>LF

Example 5:

Send: `TSP?SP2SP1LF`

Receive: `2=-1158.4405SPLF`

`1=+0000.0000LF`

INFORMATION

A maximum of 256 characters are permitted per command line.

8.3 Command Overview

Command	Format	Description
#3 (p. 142)	#3	Get Real Position
#4 (p. 143)	#4	Request Status Register
#5 (p. 143)	#5	Request Motion Status
#6 (p. 144)	#6	Query for Position Change
#7 (p. 144)	#7	Request Controller Ready Status
#8 (p. 145)	#8	Query if Macro is Running
#9 (p. 145)	#9	Get Wave Generator Status
#11 (p. 145)	#11	Get Memory Space for Trajectory Points
#24 (p. 146)	#24	Stop All Axes

Command	Format	Description
#27 (p. 146)	#27	System Abort
*IDN? (p. 147)	*IDN?	Get Device Identification
AAP (p. 147)	AAP <AxisID> <Distance> <AxisID> <Distance> ["SA" <StepSize>] ["N" <NumberOfRepetitions>] ["A" <AnalogInputID>]	Automated Alignment Part
ADD (p. 150)	ADD <Variable> <FLOAT1> <FLOAT2>	Add and Save to Variable
CCL (p. 151)	CCL <Level> [<PSWD>]	Set Command Level
CCL? (p. 152)	CCL?	Get Command Level
CPY (p. 152)	CPY <Variable> <CMD?>	Copy into Variable
CST (p. 153)	CST {<AxisID> <StageName>}	Set Assignment of Stages to Axes
CST? (p. 154)	CST? [{<AxisID>}]	Get Assignment Of Stages To Axes
CSV? (p. 154)	CSV?	Get Current Syntax Version
DEL (p. 154)	DEL <uint>	Delay the Command Interpreter
DIA? (p. 154)	DIA? [{<MeasureID>}]	Get Diagnosis Information
DIO (p. 155)	DIO {<DIOID> <OutputOn>}	Set Digital Output Lines
DIO? (p. 156)	DIO? [{<DIOID>}]	Get Digital Input Lines
DPA (p. 156)	DPA <Pswd> [{<ItemID> <PamID>}]	Reset Volatile Memory Settings To Default
DRC (p. 157)	DRC {<RecTableID> <Source> <RecOption>}	Set Data Recorder Configuration
DRC? (p. 159)	DRC? [{<RecTableID>}]	Get Data Recorder Configuration
DRL? (p. 160)	DRL? [{<RecTableID>}]	Get Number Of Recorded Points
DRR? (p. 160)	DRR? [<StartPoint> <NumberOfPoints> [{<RecTableID>}]]	Get Recorded Data Values
DRT (p. 162)	DRT {<RecTableID> <TriggerSource> <Value>}	Set Data Recorder Trigger Source
DRT? (p. 163)	DRT? [{<RecTableID>}]	Get Data Recorder Trigger Source
ECO? (p. 164)	ECO? <String>	Echo a String
ERR? (p. 164)	ERR?	Get Error Number
FDG	FDG <routine name> <scan axis> <step axis> [ML <stop level>] [A <alignment signal input channel>] [MIA <min radius>] [MAA <max radius>] [F <frequency>] [SP <speed factor>] [V <max velocity>] [MDC <max direction changes>] [SPO <speed offset>]	Fast Alignment: Defines a fast alignment gradient search routine. For details, refer to E712T0016 "Fast Multi-Channel Photonics Alignment (FMPA)".

Command	Format	Description
FDR	FDR <routine name> <scan axis> <scan axis range> <step axis> <step axis range> [L <threshold level>] [A <alignment signal input channel>] [F <frequency>] [V <velocity>] [MP1 <scan axis middle position>] [MP2 <step axis middle position>] [TT <target type>] [CM <estimation method>] [MIIL <minimum level of fast alignment input>] [MAIL <maximum level of fast alignment input>] [ST <stop position option>]	Fast Alignment: Defines a fast alignment area scanning routine. For details, refer to E712T0016 "Fast Multi-Channel Photonics Alignment (FMPA)".
FGC	FGC {<routine name> <scan axis center position> <step axis center position>}	Fast Alignment: Changes the center position of a gradient search routine that is currently running. For details, refer to E712T0016 "Fast Multi-Channel Photonics Alignment (FMPA)".
FGC?	FGC? [{<routine name>}]	Fast Alignment: Queries the current center position of a gradient search routine. For details, refer to E712T0016 "Fast Multi-Channel Photonics Alignment (FMPA)".
FIO (p. 165)	FIO <AxisID> <Distance> <AxisID> <Distance> ["S" <LinearSpiralStepSize>] ["AR" <AngularScanSize>] ["L" <Threshold>] ["A" <AnalogInputID>]	Fast Input-Output Alignment Procedure
FLM (p. 168)	FLM <AxisID> <Distance> ["L" <Threshold>] ["A" <AnalogInputID>] ["D" <ScanDirection>]	Fast Line Scan to Maximum
FLS (p. 171)	FLS <AxisID> <Distance> ["L" <Threshold>] ["A" <AnalogInputID>] ["D" <ScanDirection>]	Fast Line Scan
FRF (p. 174)	FRF [{<AxisID>}]	Fast Reference Move To Reference Switch
FRF? (p. 176)	FRF? [{<AxisID>}]	Get Referencing Result
FRH?	FRH?	Fast Alignment: Lists descriptions and physical units for the routine results that can be queried with the FRR? command For details, refer to E712T0016 "Fast Multi-Channel Photonics Alignment (FMPA)".
FRP	FRP {<routine name> <routine action>}	Fast Alignment: Stops a fast alignment routine. For details, refer to E712T0016 "Fast Multi-Channel Photonics Alignment (FMPA)".

Command	Format	Description
FRP?	FRP? [{<routine name>}]	Fast Alignment: Queries the current state of a fast alignment routine For details, refer to E712T0016 "Fast Multi-Channel Photonics Alignment (FMPA)".
FRR?	FRR? [<routine name> [<result ID>]]	Fast Alignment: Queries the results of a fast alignment routine For details, refer to E712T0016 "Fast Multi-Channel Photonics Alignment (FMPA)".
FRS	FRS {<routine name>}	Fast Alignment: Starts a fast alignment routine For details, refer to E712T0016 "Fast Multi-Channel Photonics Alignment (FMPA)".
FSA (p. 176)	FSA <Axis1ID> <Distance1> <Axis2ID> <Distance2> ["L" <Threshold>] ["S" <ScanLineDistance>] ["SA" <StepSize>] ["A" <AnalogInputID>]	Fast Scan with Automated Alignment
FSC (p. 180)	FSC <Axis1ID> <Distance1> <Axis2ID> <Distance2> ["L" <Threshold>] ["S" <ScanLineDistance>] ["A" <AnalogInputID>]	Fast Scan with Abort
FSM (p. 184)	FSM <Axis1ID> <Distance1> <Axis2ID> <Distance2> ["L" <Threshold>] ["S" <ScanLineDistance>] ["A" <AnalogInputID>]	Fast Scan to Maximum
FSS? (p. 187)	FSS?	Get Status of Fast Scan Routines
GWD? (p. 188)	GWD? [<StartPoint> <NumberOfPoints> [{<WaveTableID>}]]	Get Wave Table Data
HDI? (p. 189)	HDI?	Get Help For Interpretation Of DIA? Response
HDR? (p. 189)	HDR?	Get All Data Recorder Options
HIB? (p. 191)	HIB? [{<HIDeviceID> <HIDeviceButton>}]	Get State Of HID Button
HLP? (p. 192)	HLP?	Get List of Available Commands
HLT (p. 193)	HLT [{<AxisID>}]	Halt Motion Smoothly
HPA? (p. 193)	HPA?	Get List Of Available Parameters
IFC? (p. 194)	IFC? [{<InterfacePam>}]	Get Current Interface Parameters
IFS (p. 196)	IFS <Pswd> {<InterfacePam> <PamValue>}	Set Interface Parameters as Default Values
IFS? (p. 197)	IFS? [{<InterfacePam>}]	Get Interface Parameters as Default Values
IMP (p. 198)	IMP <AxisID> <Amplitude>	Start Impulse and Response Measurement
JRC (p. 199)	JRC <Jump> <CMD?> <OP> <Value>	Jump Relatively Depending on Condition
KCP (p. 199)	KCP <CSNameSource> <CSNameCopy>	Copy Coordinate System
KEN (p. 200)	KEN <CSName>	Activate Coordinate System
KEN? (p. 201)	KEN? [{<CSName>}]	Get Active Coordinate Systems

Command	Format	Description
KET? (p. 202)	KET? [{<CSType>}]	Get Active Coordinate System Types
KLC? (p. 203)	KLC? [<CSName1>[<CSName2> [<Item1>[<Item2>]]]]	Get Properties Of Work-And-Tool Combinations
KLD (p. 205)	KLD <CSName> [{<AxisID> <Offset>}]	Define Leveling Coordinate System By Specifying Values Before defining a leveling coordinate system, it is necessary to switch to command level 1 (see CCL).
KLF (p. 207)	KLF <CSName>	Define Leveling Coordinate System At Current Position Before defining a leveling coordinate system, it is necessary to switch to command level 1 (see CCL).
KLN (p. 209)	KLN <ChildCS> <ParentCS>	Link Coordinate Systems
KLN? (p. 210)	KLN? [{<CSName>}]	Get Coordinate System Chains
KLS? (p. 211)	KLS? [<CSName>[<Item1>[<Item2>]]]	Get Coordinate System Properties
KLT? (p. 213)	KLT? [<StartCS> [<EndCS>]]	Get Offsets Resulting From A Chain
KRM (p. 214)	KRM <CSName>	Remove Coordinate System
KSB (p. 215)	KSB <CSName> [{<AxisID> <Angle>}]	Define Orientational Coordinate System Before defining an orientational coordinate system, it is necessary to switch to command level 1 (see CCL).
KSD (p. 216)	KSD <CSName> [{<AxisID> <Offset>}]	Define Operating Coordinate System By Specifying Values
KSF (p. 218)	KSF <CSName>	Define Operating Coordinate System At Current Position
KST (p. 219)	KST <CSName> [{<AxisID> <Offset>}]	Define “Tool” Operating Coordinate System
KSW (p. 221)	KSW <CSName> [{<AxisID> <Offset>}]	Define “Work” Operating Coordinate System
LIM? (p. 223)	LIM? [{<AxisID>}]	Indicate Limit Switches
MAC (p. 223)	MAC <keyword> {<parameter>}	Call Macro Function
MAC? (p. 226)	MAC? [<macroname>]	List Macros
MAN? (p. 226)	MAN? {<CMD>}	Get Help String For Command
MEX (p. 227)	MEX <CMD?> <OP> <value>	Stop Macro Execution due to Condition
MOV (p. 228)	MOV {<AxisID> <Position>}	Set Target Position
MOV? (p. 229)	MOV? [{<AxisID>}]	Get Target Position
MRI (p. 230)	MRI {<AxisID> <Distance>}	Set Target Relative In Tool Coordinate System
MRW (p. 231)	MRW {<AxisID> <Distance>}	Set Target Relative In Work Coordinate System
MVR (p. 232)	MVR {<AxisID> <Distance>}	Set Target Relative To Current Position

Command	Format	Description
NAV (p. 233)	NAV {<AnalogInputID> <NumberOfReadings>}	Set Number of Readings to be Averaged?
NAV? (p. 234)	NAV? [{<AnalogInputID>}]	Get Number of Readings to be Averaged?
NLM (p. 234)	NLM {<AxisID> <LowLimit>}	Set Low Position Soft Limit
NLM? (p. 235)	NLM? [{<AxisID>}]	Get Low Position Soft Limit
ONT? (p. 236)	ONT? [{<AxisID>}]	Get On-Target State
PLM (p. 236)	PLM {<AxisID> <HighLimit>}	Set High Position Soft Limit
PLM? (p. 237)	PLM? [{<AxisID>}]	Get High Position Soft Limit
POS? (p. 238)	POS? [{<AxisID>}]	Get Real Position
PUN? (p. 238)	PUN? [{<AxisID>}]	Get Axis Unit
RBT (p. 239)	RBT	Reboot System
RMC? (p. 239)	RMC?	List Running Macros
RON? (p. 240)	RON? [{<AxisID>}]	Get Reference Mode
RTR (p. 240)	RTR <RecordTableRate>	Set Record Table Rate
RTR? (p. 241)	RTR?	Get Record Table Rate
SAI? (p. 241)	SAI? [ALL]	Get List Of Current Axis Identifiers
SCT (p. 242)	SCT "T" <CycleTime>	Set Cycle Time
SCT? (p. 242)	SCT? [<T>]	Get Cycle Time
SGA (p. 243)	SGA {<AnalogInputID> <Gain>}	Set Gain
SGA? (p. 243)	SGA? [{<AnalogInputID>}]	Get Gain
SIC	SIC <FA input channel ID> <calculation type> [{<calculation parameter>}]	Fast Alignment: Defines calculation settings for an analog input channel For details, refer to E712T0016 "Fast Multi-Channel Photonics Alignment (FMPA)".
SIC?	SIC? [{<FA input channel ID>}]	Fast Alignment: Queries the calculation settings for an analog input channel For details, refer to E712T0016 "Fast Multi-Channel Photonics Alignment (FMPA)".
SPA (p. 244)	SPA {<ElementID> <PamID> <PamValue>}	Set Volatile Memory Parameters
SPA? (p. 245)	SPA? [{<ElementID> <PamID>}]	Get Volatile Memory Parameters
SPI (p. 245)	SPI {<PPCoordinate> <Position>}	Set Pivot Point
SPI? (p. 247)	SPI? [{<PPCoordinate>}]	Get Pivot Point
SRG? (p. 247)	SRG? {<ItemID> <RegisterID>}	Query Status Register Value
SSL (p. 248)	SSL {<AxisID> <SoftLimitsOn>}	Set Soft Limit
SSL? (p. 249)	SSL? [{<AxisID>}]	Get Soft Limit Status
SSN? (p. 249)	SSN?	Get Device Serial Number
SST (p. 249)	SST {<AxisID> <StepSize>}	Set Step Size

Command	Format	Description
SST? (p. 250)	SST? [{<AxisID>}]	Get Step Size
STA? (p. 251)	STA?	Query Status Register Value
STE (p. 251)	STE <AxisID> <Amplitude>	Start Step And Response Measurement
STP (p. 252)	STP	Stop All Axes
SVO (p. 252)	SVO {<AxisID> <ServoState>}	Set Servo Mode
SVO? (p. 253)	SVO? [{<AxisID>}]	Get Servo Mode
TAC? (p. 254)	TAC?	Tell Analog Channels
TAD? (p. 254)	TAD? [{<InputSignalID>}]	Get ADC Value Of Input Signal
TAV? (p. 254)	TAV? [{<AnalogInputID>}]	Get Analog Input Voltage
TCI?	TCI? [{<FA input channel ID>}]	Fast Alignment: Determines calculated value of an analog input channel For details, refer to E712T0016 "Fast Multi-Channel Photonics Alignment (FMPA)".
TMN? (p. 255)	TMN? [{<AxisID>}]	Get Minimum Commandable Position
TMX? (p. 256)	TMX? [{<AxisID>}]	Get Maximum Commandable Position
TNR? (p. 257)	TNR?	Get Number Of Record Tables
TRA? (p. 257)	TRA? {<AxisID> <Component>}	Get Maximum Commandable Position For Direction Vector
TRS? (p. 258)	TRS? [{<AxisID>}]	Indicate Reference Switch
TWG? (p. 259)	TWG?	Get Number of Wave Generators
VAR (p. 259)	VAR <Variable> <String>	Set Variable Value
VAR? (p. 260)	VAR? [{<Variable>}]	Get Variable Value
VEL (p. 261)	VEL {<AxisID> <Velocity>}	Set Closed-Loop Velocity
VEL? (p. 261)	VEL? [{<AxisID>}]	Get Closed-Loop Velocity
VER? (p. 261)	VER?	Get Version
VLS (p. 262)	VLS <SystemVelocity>	Set System Velocity
VLS? (p. 263)	VLS?	Get System Velocity
VMO? (p. 263)	VMO? {<AxisID> <Position>}	Virtual Move
VST? (p. 264)	VST?	Get Connectable Stages
WAC (p. 264)	WAC <CMD?> <OP> <value>	Wait for Condition
WAV (p. 265)	WAV <WaveTableID> <AppendWave> <WaveType> <WaveTypeParameters>	Set Waveform Definition
WAV? (p. 270)	WAV? [{<WaveTableID> <WaveParameterID>}]	Get Waveform Definition
WCL (p. 271)	WCL {<WaveTableID>}	Clear Wave Table Data
WGC (p. 271)	WGC {<WaveGenID> <Cycles>}	Set Number Of Wave Generator Cycles

Command	Format	Description
WGC? (p. 272)	WGC? [{<WaveGenID>}]	Get Number Of Wave Generator Cycles
WGO (p. 272)	WGO {<WaveGenID> <StartMode>}	Set Wave Generator Start/Stop Mode
WGO? (p. 273)	WGO? [{<WaveGenID>}]	Get Wave Generator Start/Stop Mode
WGR (p. 274)	WGR	Starts Recording In Sync With Wave Generator
WGS? (p. 274)	WGS? [<WaveGenID> [<ItemID>]]	Get Status Information of Wave Generator
WMS? (p. 275)	WMS? [{<WaveTableID>}]	Get Maximum Number of Values for the Waveform
WPA (p. 276)	WPA <Pswd> [{<ElementID> <PamID>}]	Save Parameters To Nonvolatile Memory
WSL (p. 278)	WSL {<WaveGenID> <WaveTableID>}	Set Connection Of Wave Table To Wave Generator
WSL? (p. 278)	WSL? [{<WaveGenID>}]	Get Connection Of Wave Table To Wave Generator
WTR (p. 278)	WTR {<WaveGenID> <WaveTableRate> <InterpolationType>}	Set Wave Generator Table Rate
WTR? (p. 281)	WTR? [{<WaveGenID>}]	Get Wave Generator Table Rate

8.4 Command Descriptions for GCS 2.0

#3 (Get Real Position)

Description: Gets the current axis position.

Format: #3 (single ASCII character number 3)

Arguments: None

Response: {<AxisID>"="<float> LF}

where

<float> is the current axis position in physical units.

Notes: This command is identical in function to POS? (p. 238) except that only one character must be sent via the interface.

The current position of axes X, Y, Z, U, V and W is calculated from the measured positions of the individual struts.

Between the switching on of the controller and the referencing of the hexapod with FRF (p. 174), the current position of the hexapod and axes A and B is unknown. Nevertheless, the response to #3 gives the position value 0 for all axes.

The physical unit for specifying the axis position can be queried with PUN? (p. 238)

#4 (Request Status Register)

Description:	Requests system status information.
Format:	#4
Arguments:	None
Response:	The response is bit-encoded. See below for the individual codes.
Notes:	The response is the sum of the codes in hexadecimal format. For the description of the bits and an example, see "System Status Register" (p. 359). This command is identical in function to STA? (p. 251), but only one character has to be sent via the interface. Additional status information that **cannot** be queried with #4 and STA?: The C-887 has a status register (p. 358) for each of the struts 1 to 6 of the hexapod and for axes A and B. You can query the bits of these registers with the SRG? command (p. 247) and record with the C-887's data recorder (p. 97), record option 80 (status register of axis).

#5 (Request Motion Status)

Description:	Queries the motion status of the axes.
Format:	#5
Arguments:	None
Response:	The response <uint> is bit-mapped and returned as the hexadecimal sum of the following codes: 1=First axis in motion 2=Second axis in motion 4=Third axis in motion ... 0 indicates that all axes have finished moving.
Note:	The motion status of *all* hexapod axes (X to W) is "in motion", as long as at least *one* hexapod strut is in motion (strut status according to the status register bits, for details, see "Motion Status, Settling Window, Settling Time" (p. 42)).

Axes 1 to 8 correspond to the X, Y, Z, U, V, W, A and B axes in this order.

Exception: When the "NOSTAGE" positioner type is assigned to an axis (possible for axes A and B; query with the CST? command (p. 154)), this axis is not included in the bit-mapped response. In this case, it is skipped when counting the axes.

#6 (Query for Position Change)

Description: Queries whether the axis positions have changed by commanding a new target position since the last position query was sent (for commanding options, see "Supported Motion Types" (p. 32)).

Format: #6 (single ASCII character number 6)

Arguments: None

Response: The answer <uint> is bit-mapped and returned as the hexadecimal sum of the following codes:

1 = Position of the first axis has changed
2 = Position of the second axis has changed
4 = Position of the third axis has changed
...

Examples: 0 indicates that no axis position has changed.

3 indicates that the positions of the first and second axis have changed.

49 indicates that the positions of axes X, U and A have changed.

Notes: When the "NOSTAGE" positioner type is assigned to an axis (possible for axes A and B; query with the CST? command (p. 154)), this axis is not included in the bit-mapped response. In this case, it is skipped when counting the axes.

It is considered a position change when a new valid target position has been specified with a command - even within a macro - by the wave generator or via a connected control unit (p. 22) (C-887.MC2 or C-887.MC) since the last POS? (p. 238) or #3 (p. 142) was sent.

#7 (Request Controller Ready Status)

Description: Asks controller for ready status (tests if controller is ready to run a new command).

Note: Use #5 (p. 143) instead of #7 to verify if motion has ended.

Format:	#7
Arguments:	None
Response:	B1h (ASCII character 177 = "±" in Windows) if controller is ready B0h (ASCII character 176 = "°" in Windows) if controller is not ready (e.g., doing a reference move)
Troubleshooting:	The response characters may appear differently in non-Western character sets or other operating systems.

#8 (Query if Macro Is Running)

Description:	Tests if a macro is running on the controller.
Format:	#8
Arguments:	None
Response:	<uint>=0 no macro is running <uint>=1 a macro is currently running

#9 (Get Wave Generator Status)

Description:	Requests the status of the wave generator(s).
Format:	#9
Arguments:	None
Response:	The <uint> response is bit-mapped and output as the hexadecimal sum of the following codes: 1 = Wave generator 1 is active, 2 = Wave generator 2 is active, 4 = Wave generator 3 is active, etc. "Active" = Wave generator output is running
Examples:	0 indicates that no wave generator is running 5 indicates that wave generators 1 and 3 are running

#11 (Get Memory Space for Trajectory Points)

Description:	Gets the free memory space for the points of the dynamics profile.
Format:	#11 (single ASCII character number 11)

Arguments:	None
Response:	<uint> is the free memory space, given as the number of the dynamics profile points.
Notes:	#11 gets the free memory space of a buffer that contains the dynamics profile points for the hexapod. One dynamics profile point corresponds to a set of target positions for the axes of the hexapod (X, Y, Z, U, V, W). The content of the buffer is only used to define the dynamics profile when the parameters 0x19001900 and 0x19001901 both have the value 1. For further information, see "Cyclic Transfer of Target Positions" (p. 36).

#24 (Stop All Axes)

Description:	Stops all axes abruptly. See the notes below for further details. Sets error code to 10. This command is identical in function to STP (p. 252), but only one character is sent via the interface.
Format:	#24
Arguments:	None
Response:	None
Notes:	#24 stops all axis motion caused by motion commands, fast alignment routines, scanning procedures or wave generator output, and stops the reference move. #24 stops macros. After the axes are stopped, their target positions are set to their current positions.

#27 (System Abort)

Description:	Stops the controller.
Format:	#27 (single ASCII character number 27)
Arguments:	None
Response:	None

Notes: #27 triggers the following:

- Motors of the struts are switched off
- Servo mode is switched off
- Commands are no longer processed
- Parameters in the volatile memory are set to default values
- Power adapter of the controller remains switched on

For rebooting, the controller must be switched off and back on again.

***IDN? (Get Device Identification)**

Description: Reports the device identity number.

Format: *IDN?

Arguments: None

Response: Single-line text terminated with a termination character (line feed) with controller name, serial number, and firmware version

Note: Example of a response for the C-887:

```
©2011-2017 Physik Instrumente (PI) GmbH & Co.
KG,C-887,117009333,2.3.1.1
```

Meaning of the response:

- `C-887` device name
- `117009333` Serial number of the C-887
- FW: `2.3.1.1`

AAP (Automated Alignment Part)

Description: Starts a scanning procedure for a detailed determination of the maximum intensity of an analog input signal.

The scanning procedure started with AAP corresponds to the "fine portion" of the scanning procedure that was started with the FSA command (p. 176).

Scans a specified area ("scanning area") using a gradient formation in the intensity until a local maximum of the analog input signal is found.

The plane in which the scanning area is located can be defined by one of the following axis pairs:

X Y

Y Z

X Z

U V

U W

V W

The scanning procedure starts at the current position at the time the command is received ("initial position"). The scanning area is centered around the initial position.

The step size of the motion is automatically adjusted in order to detect the local maximum as precisely as possible.

AAP (and therefore the motion of the platform) has finished successfully when the number of checks specified with <NumberOfRepetitions> has confirmed that the local maximum is at the current position. The influence of a superimposed noise signal can be reduced by multiple checks.

When <NumberOfRepetitions> is set to zero, a gradient of the intensity is continually sought. With this setting, a moving local maximum can be continuously tracked. The scanning procedure does not end until it is aborted with #24 (p. 146), STP (p. 252) or HLT (p. 193) and is then considered to be unsuccessfully completed.

AAP was finished unsuccessfully in the following cases:

- No gradient of the intensity was found: the motion platform returns to the initial position.
- The scanning area would be exceeded: the motion platform returns to the initial position.
- #24 (p. 146), STP (p. 252) or HLT (p. 193) was sent: The motion platform remains in the current position

FSS? (p. 187) can be used to check whether a scanning procedure has been successfully completed.

In order to check whether a scanning procedure is still going on, the motion status of the axes can be queried with #5 (p. 143).

Format: AAP <AxisID> <Distance> <AxisID> <Distance>
["SA" <StepSize>] ["N" <NumberOfRepetitions>]
["A" <AnalogInputID>]

Arguments: <AxisID>: Is one axis of the controller, format: string
Axes X, Y, Z, U, V, W are permissible.

<Distance>: distance to be scanned along the axis, format: double

"SA": Required keyword for entering <StepSize>
<StepSize>: Starting value for the step size, format: double

"N": Required keyword for entering <NumberOfRepetitions>
<NumberOfRepetitions>: Number of successful checks of the local maximum at the current position that is required for finishing AAP successfully. Format: integer

"A": Required keyword for entering <AnalogInputID>
<AnalogInputID>: is the identifier of the analog input signal whose maximum intensity is sought, format: Integer

Response: None

Notes: The physical unit for specifying the <Distance> and <Stepsize> can be queried with PUN?

The values for <Distance> must be identical for both axes.

The following default values are used when the corresponding arguments are omitted:
<StepSize>: 0.001
<NumberOfRepetitions>: 3
<AnalogInputID>: 1
These default values have only been checked for the H-810, H-811, and H-206 hexapods. For other hexapods, the ideal parameters must be determined experimentally.

If there are several local maxima in the scanning area, FSM (p. 184) can be used to find the global maximum intensity before AAP is called. It is recommended to do an FSM scanning procedure with a large step size over the entire scanning area, first and then to scan a smaller area with FSM using a smaller step size, and finally to optimize alignment of the motion platform with the global maximum intensity found with AAP.

The greater the noise of the signal to be scanned, the more frequently it should be checked whether a local maximum has been reached, i.e., a large value should be selected for <NumberOfRepetitions> if possible.

The greater the noise of the signal to be scanned, the higher the number of readout values of the analog signal should be set with NAV (p. 233) for averaging.
At the same time, it should be noted that this setting increases the time required for the scanning procedure.

When the ***Trajectory Source*** parameter (ID 0x19001900) is set to 1, the dynamics profile must be defined by consecutive MOV commands, and AAP is not allowed.
For more information, see "Hexapod Motion" (p. 31).

Example: Send: AAP Y 0.1 Z 0.1 SA 0.001 N 3 A 2

Starts a scanning procedure in the YZ plane:

- Side length of the square area: 0.1 mm
- Start value of the step size: 1 µm
- Number of successful checks of the local maximum after which AAP has finished successfully: 3
- Identifier of the analog input channel whose maximum intensity is sought: 2

ADD (Add and Save to Variable)

Description: Adds two values and saves the result to a variable (p. 131).

The variable is present in volatile memory (RAM) only.

Format: ADD <Variable> <FLOAT1> <FLOAT2>

Arguments: <Variable> is the name of the variable to which the result is to be saved.

<FLOAT1> is the first summand.

<FLOAT2> is the second summand.

Floating point numbers are expected for the summands. They can be specified directly or via the value of a variable.

Response: None

Note: ADD can only be used in macros.

Example 1: Value $B is added to value $A, and the result is saved to variable C:

Send: `ADD C $A $B`

Example 2: The name of the variable to which the result is to be copied is given via the value of another variable:

Send: `VAR?`
Receive: `A=468`
`B=123`
`3Z=WORKS`
Send: `ADD A${3Z} $A $B`
Send: `VAR?`
Receive: `A=468`
`B=123`
`AWORKS=591`
`3Z=WORKS`
Send: `ADD ${3Z} $A $B`
Send: `VAR?`
Receive: `A=468`
`B=123`
`AWORKS=591`
`WORKS=591`
`3Z=WORKS`

CCL (Set Command Level)

Description: Changes the active "command level" and therefore determines the availability of commands and write access to system parameters.

Format: CCL <Level> [<PSWD>]

Arguments: <Level> is a command level of the controller

<PSWD> is the password required for changing to the appropriate command level

The following command levels and passwords apply:

Level = 0 is the default setting, all commands provided for "normal" users are available, read access to all parameters, no password required.

Level = 1 adds additional commands and write access to level-1 parameters (commands and parameters from level 0 are included). The password required is "advanced".

Level > 1 is provided for PI service personnel only. Users cannot change to a level > 1. Contact the customer service department if you have problems with parameters for command level 2 or higher (p. 343).

Response: None

Troubleshooting: Invalid password

Notes: HLP? (p. 192) lists all commands available in the current command level.

HPA? (p. 193) lists the parameters including the information about which command level allows write access to them. For more information about using parameters, see "Adjusting Settings" (p. 299).

After controller power-on or reboot, the active command level is always level 0.

CCL? (Get Command Level)

Description: Get the active "command level".

Format: CCL?

Arguments: none

Response: <Level> is the currently active command level; uint.

Notes: <Level> should be 0 or 1.

<Level> = 0 is the default setting, all commands provided for "normal" users are available, as is read access to all parameters

<Level> = 1 adds additional commands and write access to level 1 parameters (commands and parameters from level 0 are included).

CPY (Copy Into Variable)

Description: Copies a command response to a variable (p. 131).

The variable is present in volatile memory (RAM) only.

Format: CPY <Variable> <CMD?>

Arguments: <Variable> is the name of the variable to which the command response is to be copied.

<CMD?> is one query command in its usual notation. The response has to be a single value and not more.

Response: None

Note: CPY can only be used in macros.

Example: It is possible to copy the value of one variable (e.g. SOURCE) to another variable (e.g. TARGET):

Send: `CPY TARGET VAR? SOURCE`

CST (Set Assignment of Stages to Axes)

Description: Assign a positioner type to an axis.

Format: CST {<AxisID> <StageName>}

Arguments: <AxisID> is one axis of the controller
<StageName> is the name of the positioner type

Notes: Assigning a positioner type with the CST command is only permissible for axes A and B.

CST also switches servo mode on for axes A and B.

CST loads the operating parameters from the PIStages2.dat positioner database for the assigned positioner type on the controller into the working memory. The permissible positioner types can be listed with the VST? command (p. 264). You can find further information in "Operating Parameters for Axes A and B" (p. 30).

The assignment of the positioner types can be saved in the nonvolatile memory of the C-887 with the WPA A12 command (p. 276).

The current assignment of positioner types can be queried with the CST? command. (p. 154)

The axis is "deactivated" if the positioner type is set to NOSTAGE with the CST command. A deactivated axis is not accessible for axis-related commands (e.g., motion commands or position queries). The identifier of a deactivated axis can only be queried with SAI? ALL.

CST? (Get Assignment Of Stages To Axes)

Description:	Returns the name of the connected positioner type for the queried axis.
Format:	CST? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller
Response:	{<AxisID>"="<string> LF} where <string> is the name of the positioner type assigned to the axis.

CSV? (Get Current Syntax Version)

Description:	Queries the GCS syntax version used in the firmware.
Format:	CSV?
Arguments:	None
Response:	The current GCS syntax version
Note:	The only possible response is 2.0 (for GCS 2.0).

DEL (Delay the Command Interpreter)

Description:	Delays <uint> milliseconds.
Format:	DEL <uint>
Arguments:	<uint> is the delay value in milliseconds.
Response:	None
Note:	DEL should only be used in macros.

DIA? (Get Diagnosis Information)

Description:	Gets the current value of the given measurand. If all arguments are omitted, the current values of all measurands are queried.

Format: DIA? [{<MeasureID>}]

Arguments: <MeasureID> is the identifier of one measurand; see below for details.

Response: {<MeasureID>"="<MeasuredValue> LF}

where

<MeasuredValue> gives the current value of the measurand; see below for details.

Notes: Use the response to the HDI? command (p. 189) to obtain descriptions and physical units of the supported measurands.

C-887 supports the following measurands:

<MeasureID>	<Description> (get with HDI?)
1	Hexapod Powered: Power Good signal (pin 59 of the **Hexapod** socket (p. 354)): 0 = Power Good signal outside of the required range or not present. 1 = Power Good signal within the required range Note: The value of the ***Check PowerGood Signal*** parameter (ID 0x19004000) is **not** relevant for the response to DIA?.
2	Controller E-Stop Activated: Current status 0 = OFF, i.e., 24 V output deactivated, no Power OK signal 1 = ON, i.e., 24 V output activated, Power OK signal is available
3	Temperature of Controller: Current temperature of the CPU, in °C.
4	Index of Faulty Point in Waveform: Index of the waveform point at which the last error occurred during wave generator output

DIO (Set Digital Output Line)

Description: Switches the specified digital output line(s) to specified state(s).

Use TIO? to get the number of installed digital I/O lines.

Format: DIO {<DIOID> <OutputOn>}

Arguments:	<DIOID> is one digital output line of the controller, see below for details. <OutputOn> is the state of the digital output line, see below for details.
Response:	none
Notes:	Using the DIO command, you can activate/deactivate digital output lines 1 to 4, which are located on the **I/O** socket (p. 354). The <DIOID> identifiers to be used for the lines are 1 to 4. If <OutputOn>=1 the line is set to HIGH/ON; if <OutputOn>=0 it is set to LOW/OFF.

DIO? (Get Digital Input Lines)

Description:	Queries the states of the specified digital input lines. Use TIO? to query the number of available digital I/O lines.
Format:	DIO? [{<DIOID>}]
Arguments:	<DIOID> is the identifier of the digital input line, see below for details.
Response:	{<DIOID>"="<InputOn> LF} where <InputOn> specifies the state of the digital input line, see below for details.
Notes:	You can use the DIO? command to directly read the digital input lines 1 to 4 that are located on the **I/O** socket (p. 354). The <DIOID> identifiers to be used for the lines are 1 to 4. If the identifier is omitted, all lines are queried. If <InputOn>=0, the digital input signal is LOW/OFF; if <InputOn>=1, the digital input signal is HIGH/ON.

DPA (Reset Settings to Default)

Description:	Resets parameter values and parameter-independent settings to the default settings.
Format:	DPA <Pswd> [{<ItemID> <PamID>}]

Arguments:	<Pswd> is the password for resetting the memory. See below for details. <ElementID> is the element for which a parameter is to be reset. See below for details. <PamID> is the parameter identifier, can be written in hexadecimal or decimal format. See below for details.
Response:	None
Troubleshooting:	Illegal element identifier, wrong parameter ID, invalid password
Notes:	DPA resets the parameter values and the settings for the coordinate systems stored in the volatile memory of the C-887 to the default settings. For the C-887, DPA is not necessary for the specification of <ItemID> and <PamID>. The default settings loaded using DPA are independent of the settings in the nonvolatile memory, which can be overwritten using WPA (p. 276). The settings in the nonvolatile memory (saved with WPA) are loaded to the volatile memory automatically when the C-887 is switched on or rebooted.

Valid passwords:

100	Resets the values of all parameters and the settings for coordinate systems to the default settings (for details, see password SKS)
SKS	Default settings which are restored using DPA SKS: ▪ Orientational coordinate system (KSB() type): PI_BASE is active ▪ Leveling coordinate system (KLD() or KLF() type): PI_Levelling is active ▪ ZERO operating coordinate system is active and based on PI_BASE and PI_Levelling ▪ Pivot point (see SPI (p. 245)), soft limits of axes (see NLM (p. 234), PLM (p. 236), and SSL (p. 248)), step size for motion initiated by a manual control unit (see SST (p. 249))

DRC (Set Data Recorder Configuration)

Description:	Determines the data source to be used and the type of data to be recorded (record option) for the data recorder table specified.
Format:	DRC {<RecTableID> <Source> <RecOption>}
Arguments:	<RecTableID> is one data recorder table of the controller, see below.

<Source> is the ID of the data source, for example, an axis or channel of the controller. The required source depends on the selected record option.

<RecOption> is the type of data to be recorded (record option).

Refer to the following list of available record options and the corresponding data sources for details

Response: None

Notes: The number of available data recorder tables can be read with TNR? (p. 257). The C-887 has 16 data recorder tables.

The maximum number of points per data recorder table available for data recording is 262144 (default: 8192).

With HDR? (p. 189) you will obtain a list of available record options and information on additional parameters and commands concerned with data recording.

For more information see "Data Recording" (p. 97).

Record options for the corresponding data sources:

<Source>	<RecOption>
Axis / hexapod strut	0 = Nothing is recorded 1 = Commanded position of axis 2 = Real position of axis 3 = Position error of axis 8 = Measurement time 70 = Commanded velocity of axis 71 = Commanded acceleration of axis 73 = Motor output of axis 74 = Current proportion to the current error of axis 75 = Current integrated position error of axis 76 = Current derivative position error of axis 80 = Status register of axis Further information can be found in "Status Registers for Hexapod Struts and Axes A and B" (p. 358). 84=Real position of second sensor in axis 86 = Fifo size for continuous position mode 87 = Commanded position for continuous position mode Continuous position mode stands for the "cyclic transfer of target positions" (p. 36); the values that have already been processed by C-887 are recorded, and not the received values.

Input signal channel	17 = Input value of channel, calculated, in volts. 18 = Input value of channel, directly from channel, without dimension. 150 = Input value of channel, calculated in volts, input calculation made with SIC

HDR? indicates for which data sources the individual record options can be used. For details on existing axes and channels, see "Commandable Elements" (p. 25).

The defaults of the data recorder are as following:

```
drc?
1=X 1
2=X 2
3=Y 1
4=Y 2
5=Z 1
6=Z 2
7=U 1
8=U 2
9=V 1
10=V 2
11=W 1
12=W 2
13=1 8
14=0 0
15=0 0
16=0 0
```

Example: Send: DRC 4 X 2

The current position of axis X is to be recorded in data recorder table 4.

DRC? (Get Data Recorder Configuration)

Description: Queries the settings for the data to be recorded.

Format: DRC? [{<RecTableID>}]

Arguments: <RecTableID>: is a data recorder table of the controller; if this entry is not specified, the response will contain the settings for all tables.

Response:	The current DRC settings: {<RecTableID>"="<Source> <RecOption> LF} where <Source>: is the data source, for example, an axis or a channel of the controller. The source type depends on the record option. <RecOption>: is the type of data to be recorded (record option). The available record options can be queried with HDR? (p. 189).

DRL? (Get Number of Recorded Points)

Description:	Reads the number of points comprised by the last recording.
Format:	DRL? [{<RecTableID>}]
Arguments:	<RecTableID> is one data recorder table of the controller
Response:	{<RecTableID>"="<uint> LF} where <uint> specifies the number of points recorded with the last recording
Notes:	The number of points is reset to zero for a data recorder table when changing its configuration with DRC (p. 157).

DRR? (Get Recorded Data Values)

Description:	Queries the last recorded data. Querying can take some time depending on the number of points to be read! It is possible to read the data while recording is still in progress.
Format:	DRR? [<StartPoint> <NumberOfPoints> [{<RecTableID>}]]
Arguments:	<StartPoint> is the first point to be read from the data recorder table, starts with index 1. <NumberOfPoints> is the number of points to be read per table. <RecTableID> is one data recorder table of the controller.

Response: For the recorded data in GCS array format, refer to the separate manual for the GCS array, SM146E, and the example below.

Notes: If the value -1 is set for <NumberOfPoints>, all valid points of the selected tables will be read out

If <RecTableID> is omitted, all tables whose record option set with DRC (p. 157) is not equal to zero are read out.

With HDR? (p. 189), you will obtain a list of all available record and trigger options as well as information on additional parameters and commands for data recording.

For further information, see "Data Recorder" (p. 97).

Example:

```
drc 1 X 1 2 X 2
drt 1 1 1
mov × 2
drr? 1 5 1 2 13
# REM data recorded with C-887 controller
#
# TYPE = 1
# SEPARATOR = 32
# DIM = 3
# NDATA = 5
# SAMPLE_TIME = 0.001000
#
# NAME0 = TARGET POSITION X
# NAME1 = REAL POSITION X
# NAME2 = MEASUREMENT TIME Strut 1
# DISPUNIT0 = mm
# DISPUNIT1 = mm
# DISPUNIT2 = sec
# END_HEADER
0.004220971838 -0.000364157866 0
0.064530499279 0.036015000194 0.005040000193
0.145174950361 0.140960231423 0.010040000081
0.232607111335 0.216114446521 0.015200000256
0.309781700373 0.30354321003 0.020020000637
```

DRC (p. 157) is used to determine that the current position of axis X is to be recorded in data recorder table 1 and that the commanded position of axis X is to be recorded in data recorder table 2.

DRT (p. 162) is used to determine that a recording is to be triggered by the next motion command, e.g., MOV (p. 228).

MOV triggers the motion of axis X to position 2.

DRR? is used to get the first five points of data recorder tables 1, 2 and 13.

The default setting of the data recorder is that the time is recorded in data recorder table 13.

DRT (Set Data Recorder Trigger Source)

Description:	Defines a trigger source for the specified data recorder table.
Format:	DRT <RecTableID> <TriggerSource> <Value>
Arguments:	<RecTableID> is one data recorder table of the controller. See below for details. <TriggerSource> ID of the trigger source, see below for a list of available options. <Value> depends on the trigger source, can be a dummy, see below.
Response:	none
Notes:	Regardless of the data recorder table selected with <RecTableID>, the trigger option selected with <Triggersource> is always set for all data recorder tables. With HDR? (p. 189), you will obtain a list of all available recording and triggering options as well as additional information on data recording. IMP (p. 198), STE (p. 251), WGO (p. 272), and WGR (p. 274) each trigger a recording by the data recorder regardless of the DRT settings (WGR only when the wave generator is active).

The data recorder configuration, i.e., the assignment of data sources and record options to the recorder tables, is set with DRC (p. 157). A recording is actually only started for data recorder tables whose record option set with DRC is not equal to zero.

The recording ends when the maximum number of points is reached (specified by the ***Data Recorder Points Per Table*** parameter, ID 0x16000201).

For more information see "Data Recording" (p. 97).

Available trigger options:

0 = No trigger. Exception: STE, IMP, WGO, and WGR always trigger data recording;
Data recording is not triggered, and data recording in progress is continued until the maximum number of points is reached. Exception: STE, IMP, WGO (if used to start the function generator output), and WGR always trigger a recording. <Value> must be a dummy.

1 = Trigger with next command that changes the position;
e.g. MOV (p. 228), MVR (p. 232); <Value> must be a dummy. Default setting.

2 = Trigger with next command;
sets trigger to the trigger option 0 after execution; <Value> must be a dummy.

4 = Trigger immediately;
sets trigger to the trigger option 0 after execution; <Value> must be a dummy

6 = Trigger with next command that changes the position, e.g. MOV, MVR; sets trigger to the trigger option 0 after execution; <Value> must be a dummy.

If trigger option 2, 4, or 6 is set, then trigger option 0 is automatically activated after the recording is triggered. As a result, the recording continues until the maximum number of points of the data recorder table(s) is reached. In this way, several motion can be recorded in succession. Trigger options 2, 4, or 6 are a good idea, e.g., when a dynamics profile is to be defined by consecutive MOV commands (see MOV).

DRT? (Get Data Recorder Trigger Source)

Description: Queries the trigger source for the data recorder tables.

Format: DRT? [{<RecTableID>}]

Arguments:	<RecTableID> is one data recorder table of the controller.
Response:	{<RecTableID>"="<TriggerSource> <Value> LF} where <TriggerSource> is the identifier of the trigger source. <Value> depends on the trigger source. Further information can be found in the description of the DRT command (p. 162).

ECO? (Echo a String)

Description:	Returns a string. ECO? can be used to test the communication.
Format:	ECO? <String>
Arguments:	<String> is any given combination of characters consisting of letters and numbers
Response:	<String> LF
Note:	<String> can consist of up to 100 characters.

ERR? (Get Error Number)

Description:	Get error code <int> of the last occurred error and reset the error to 0. Only the last error is buffered. You should therefore call ERR? after each command. The error codes and their descriptions are listed in "Error Codes" (p. 281).
Format:	ERR?
Arguments:	None
Response:	The error code of the last error that occurred (integer).

Troubleshooting: Communication breakdown

Notes: In the case of simultaneous access to the controller by several instances, the error code is only returned to the first instance that sent the ERR? command. Because the error is reset to 0 by the query, the error is not visible for any further querying instance.

- ➢ If possible, access the controller with one instance only.
- ➢ If incorrect system behavior does not cause the controller to send an error code, check whether the error code is queried regularly in the background by a macro, script or the PC software (e.g., PIMikroMove).

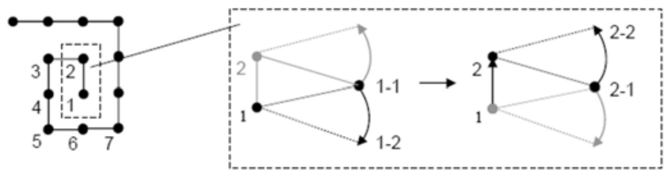
If the cause of an error continues, the corresponding error code is immediately set again after a query with ERR?.

Example: The configuration file for the connected hexapod is missing in the C-887. The error code 233 is set until a suitable configuration file is loaded to the C-887, despite the query.

FIO (Fast Input-Output Alignment Procedure)

Description: Starts a scanning procedure for the alignment of optical elements (e.g. optical fibers), the input and output of which are on the same side. Within the element, light is transmitted from the input to the output. In order for a sensor that is connected with the analog input channel of the controller to receive the signal at the output of the optical element with maximum intensity, the input and output of the optical element must be aligned at the same time.

Moves along a linear spiral in a specified area with a specified step length, see figure



The level in which the specified area is located can be defined by one of the following axis pairs:

X Z

Y Z

X Y

The scanning procedure starts at the current position at the time the command is received ("initial position").

At the start point and after each step, scanning is done at a specified angle around the pivot point until an analog input signal reaches a specified intensity threshold. This corresponds to an angular scan around the axis that is perpendicular to the scanned area.

The pivot point must be at the position that corresponds to the center of the input or the center of the output of the optical element. The coordinates of the pivot point can be set with SPI (p. 245).

FIO (and therefore the motion of the platform) has finished successfully when the analog input signal reaches the specified intensity threshold.

FIO is finished unsuccessfully in the following cases:

- The intensity threshold has not been reached in the specified area: the motion platform returns to the initial position.
- #24 (p. 146), STP (p. 252) or HLT (p. 193) was sent: The motion platform remains in the current position

FSS? (p. 187) can be used to check whether a scanning procedure has been successfully completed.

In order to check whether a scanning procedure is still going on, the motion status of the axes can be queried with #5 (p. 143).

Format: FIO <AxisID> <Distance> <AxisID> <Distance>
["S" <LinearSpiralStepSize>] ["AR" <AngularScanSize>]
["L" <Threshold>] ["A" <AnalogInputID>]

Arguments: <AxisID>: Is one axis of the controller, format: string

Axes X, Y, and Z are permissible.

<Distance>: distance to be moved along the axis, format: double

"S": Required keyword for entering <LinearSpiralStepSize>

<LinearSpiralStepSize>: Step size in which the platform moves along the spiral path, format: double

"AR": Required keyword for entering <AngularScanSize>

<AngularScanSize>: Angle around the pivot point at which scanning is done, in degrees, format: double

"L": Required keyword for entering <Threshold>

<Threshold>: Intensity threshold of the analog input signal, in V, format: Double

"A": Required keyword for entering <AnalogInputID>

<AnalogInputID>: is the identifier of the analog input signal whose maximum intensity is sought, format: Integer

Response: None

Notes: The physical unit for specifying the <Distance> and <LinearSpiralStepSize> can be queried with PUN? (p. 238).

The area specified by the values for <Distance> must be square.

The following default values are used when the corresponding arguments are omitted:

<LinearSpiralStepSize>: 0.01 mm

<AngularScanSize: 0.2 degrees

<Threshold>: 1.0 V

<AnalogInputID>: 1

These default values have only been checked for the H-810, H-811, and H-206 hexapod models. For other hexapod models, the ideal parameters must be determined experimentally.

The lower the system velocity is set with VLS (p. 262) during a scanning procedure, the greater the accuracy with which the threshold of the intensity is found.

Velocities in the range of under 1 mm/s are recommended.

Note:
If the values for distances or angles are too large, the platform moves on an undefined path during a scanning procedure and it tilt. Therefore, collisions are possible and it is also possible that the result of the scanning procedure is unsatisfactory. Measures for avoiding tilting:

- Select values of up to 0.2 degrees for <AngularScanSize>
- Align the motion platform suitably before starting the scanning procedure.
- Use suitable holders for the inputs and/or outputs of the optical element to be aligned on the motion platform so that the motion during the scanning procedure only takes place over short distances or angles.

When the ***Trajectory Source*** parameter (ID 0x19001900) is set to 1, the dynamics profile must be defined by consecutive MOV commands, and FIO is not allowed.
For more information, see "Hexapod Motion" (p. 31).

Example: FIO Y 0.1 Z 0.1 S 0.01 AR 0.1 L 1 A 2

Starts a scanning procedure along a linear spiral in the YZ level:

- Side length of the square area: 0.1 mm
- Step size: 10 µm
- Angle around the pivot point: 0.1 degrees (for axis U)
- Intensity threshold: 1 V
- Identifier of the analog input channel whose maximum intensity is sought: 2

FLM (Fast Line Scan to Maximum)

Description: Starts a scanning procedure to determine the global maximum intensity of an analog input signal.

Completely scans a specified distance along an axis for the intensity of the analog input signal. In the case of several maximum intensities, this prevents that only a local maximum is found instead of the global maximum.

The direction of the scanning procedure as well as the starting and end position of the distance can be specified with the argument <ScanDirection>.

FLM (and therefore the motion of the platform) is finished successfully when the two following conditions have been met:

- The analog input signal has reached the specified intensity threshold on the specified distance at least once.
- The motion platform has returned from the end position of the distance to the position with the maximum intensity.

FLM is finished unsuccessfully in the following cases:

- The intensity threshold has not been reached on the specified distance: the motion platform returns from the end position of the distance to the initial position (current position at the time the command is received).
- #24 (p. 146), STP (p. 252) or HLT (p. 193) was sent: The motion platform remains in the current position

FSS? (p. 187) can be used to check whether a scanning procedure has been successfully completed.

In order to check whether a scanning procedure is still going on, the motion status of the axes can be queried with #5 (p. 143).
A similar scanning procedure can be started with FLS (p. 171). In contrast to FLM, FLS has already finished successfully when the intensity threshold is reached for the first time.

Format: FLM <AxisID> <Distance> ["L" <Threshold>] ["A" <AnalogInputID>] ["D" <ScanDirection>]

Arguments: <AxisID>: Is one axis of the controller, format: string

Axes X, Y, Z, U, V, W are permissible.

<Distance>: distance to be scanned along the axis, format: double

"L": Required keyword for entering <Threshold>

<Threshold>: Intensity threshold of the analog input signal, in V, format: Double

"A": Required keyword for entering <AnalogInputID>

<AnalogInputID>: is the identifier of the analog input signal whose maximum intensity is sought, format: Integer

	"D": Required keyword for entering <ScanDirection> <ScanDirection>: Specifies the direction of the scanning procedure as well as the start and end position of the distance: 0: Scanning procedure is done centered around the current position, in positive direction. Starting position = current position - <Distance>/2 Ending position = current position + <Distance>/2 1: Scanning procedure in positive direction: Starting position = current position Ending position = current position + <Distance> -1: Scanning procedure in negative direction: Starting position = current position Ending position = current position - <Distance>
Response:	None
Notes:	The physical unit for specifying the <Distance> can be queried with PUN? (p. 238) The following default values are used when the corresponding arguments are omitted: <Threshold>: 1.0 V <AnalogInputID>: 1 <ScanDirection>: 0 (scanning procedure centered around the current position, in positive direction) The lower the system velocity is set with VLS (p. 262) during a scanning procedure, the greater the accuracy with which the maximum intensity is found. Velocities in the range of under 1 mm/s are recommended.

If the values for distances or angles are too large, the platform moves on an undefined path during a scanning procedure and it tilt. Therefore, collisions are possible and it is also possible that the result of the scanning procedure is unsatisfactory. Measures for avoiding tilting:

- Select suitable values for <Distance>. For the hexapod models H-810, H-811, and H-206, 0.2 mm or 0.2 degrees should not be exceeded; for other hexapod models as well as in the case of changed settings for coordinate systems and the pivot point, the ideal values have to be determined experimentally.
- Set the velocity for the motion platform of the hexapod as low as possible (with the VLS command).
- Align the motion platform suitably before starting the scanning procedure.
- Use suitable holders for the inputs and/or outputs of the optical element to be aligned on the motion platform so that the motion during the scanning procedure only takes place over short distances or angles.

When the ***Trajectory Source*** parameter (ID 0x19001900) is set to 1, the dynamics profile must be defined by consecutive MOV commands, and FLM is not allowed.

For more information, see "Hexapod Motion" (p. 31).

Example: FLM Z 0.2 L 1.2 A 2 D 0

Starts a scanning procedure along the Z axis:

- Distance: 0.2 mm
- Intensity threshold: 1.2 V
- Identifier of the analog input channel whose maximum intensity is sought: 2
- Scanning procedure takes place centered around the current position, in positive direction

FLS (Fast Line Scan)

Description: Starts a scanning procedure which scans a specified distance along an axis until the analog input signal reaches a specified intensity threshold.

The direction of the scanning procedure as well as the start position of the distance can be specified with the argument <ScanDirection>.

When the intensity threshold has already been reached at the initial position (current position at the time the command is received), a different start position will not be approached.

FLS (and therefore the motion of the platform) is finished successfully when the analog input signal reaches the specified intensity threshold. If necessary due to the braking distance, the platform will return to the position at which the intensity threshold was reached after stopping.

FLS is finished unsuccessfully in the following cases:

- The intensity threshold has not been reached on the specified distance: the motion platform returns from the end position of the distance to the initial position.
- #24 (p. 146), STP (p. 252) or HLT (p. 193) was sent: The motion platform remains in the current position

FSS? (p. 187) can be used to check whether a scanning procedure has been successfully completed.

In order to check whether a scanning procedure is still going on, the motion status of the axes can be queried with #5 (p. 143).

A similar scanning procedure can be started with FLM (p. 168). In contrast to FLS, FLM scans along the entire specified distance, so that the global maximum is found if there are several local maximums.

Format: FLS <AxisID> <Distance> ["L" <Threshold>] ["A" <AnalogInputID>] ["D" <ScanDirection>]

Arguments: <AxisID>: Is one axis of the controller, format: string

Axes X, Y, Z, U, V, and W are permissible.

<Distance>: distance to be scanned along the axis, format: double

"L": Required keyword for entering <Threshold>

<Threshold>: Intensity threshold of the analog input signal, in V, format: Double

"A": Required keyword for entering <AnalogInputID>

<AnalogInputID>: is the identifier of the analog input signal whose maximum intensity is sought, format: Integer

"D": Required keyword for entering <ScanDirection>

<ScanDirection>: Specifies the direction of the scanning procedure as well as the start and end position of the distance:

0: Scanning procedure is done centered around the current position, in positive direction.

Starting position = current position - <Distance>/2

Ending position = current position + <Distance>/2

1: Scanning procedure in positive direction:

Starting position = current position

Ending position = current position + <Distance>

-1: Scanning procedure in negative direction:

Starting position = current position

Ending position = current position - <Distance>

Response: None

Notes: The physical unit for specifying the <Distance> can be queried with PUN?

The following default values are used when the corresponding arguments are omitted:

<Threshold>: 1.0 V

<AnalogInputID>: 1.

<ScanDirection>: 0 (scanning procedure centered around the current position, in positive direction)

The lower the system velocity is set with VLS (p. 262) during a scanning procedure, the greater the accuracy with which the maximum intensity is found.

Velocities in the range of under 1 mm/s are recommended.

If the values for distances or angles are too large, the platform moves on an undefined path during a scanning procedure and it tilt. Therefore, collisions are possible and it is also possible that the result of the scanning procedure is unsatisfactory. Measures for avoiding tilting:

- Select suitable values for <Distance>. For the hexapod models H-810, H-811, and H-206, 0.2 mm or 0.2 degrees should not be exceeded; for other hexapod models as well as in the case of changed settings for coordinate systems and the pivot points, the ideal values have to be determined experimentally.
- Set the velocity for the motion platform of the hexapod as low as possible (with the VLS command).
- Align the motion platform suitably before starting the scanning procedure.
- Use suitable holders for the inputs and/or outputs of the optical element to be aligned on the motion platform so that the motion during the scanning procedure only takes place over short distances or angles.

When the ***Trajectory Source*** parameter (ID 0x19001900) is set to 1, the dynamics profile must be defined by consecutive MOV commands, and FLS is not allowed.

For more information, see "Hexapod Motion" (p. 31).

Example: FLS Z 0.2 L 1 A 2 D 0

Starts a scanning procedure along the Z axis:

- Distance: 0.2 mm
- Intensity threshold: 1 V
- Identifier of the analog input channel whose maximum intensity is sought: 2
- Scanning procedure takes place centered around the current position, in positive direction

FRF (Fast Reference Move To Reference Switch)

Description: Starts a reference move.

Moves the specified axis to the reference switch and sets the current position to a defined value. See below for details.

If multiple axes are specified in the command, they are started simultaneously.

Format:	FRF [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller, if left out, all axes are affected.
Response:	None
Troubleshooting:	Illegal axis identifier

Notes:

For axes with incremental sensors, motion can only be commanded after a successful reference move (also referred to as "initialization").

The behavior of the axes of the hexapod after the reference move is determined by the ***Behaviour After Reference Move*** (ID 0x07030401) and ***Target For Motion After Reference Move*** parameters (ID 0x07030402). Depending on the parameter values, the axes of the platform can be moved automatically to a specified position, e.g., after the reference move.

- Value of the parameter 0x07030401 = 0: The axis remains in the reference position after the reference move.
- Value of parameter 0x07030401 = 1: After the reference move, the axis moves to the target position which is given by parameter 0x07030402.

No reference move is required for axes with absolute-measuring sensors. The use of the FRF command is still recommended for these axes, however. FRF does not start a reference move for axes with absolute-measuring sensors but sets the target positions to the current position values. The parameter values described above also take effect, so that the axes can be moved to a defined "initial position", for example.

The hexapod moves unpredictably during a reference move. A collision check or prevention does **not** take place, even if a configuration for preventing collisions was stored on the C-887 with the PIVeriMove hexapod software for collision checking. Soft limits that have been set for the motion platform of the hexapod with the NLM (p. 234) and PLM (p. 236) commands are ignored during the reference move.

A common reference move is always done for the axes of the motion platform of the hexapod (X, Y, Z, U, V, W). It is therefore always sufficient to enter a single axis to start the reference move of the motion platform, e.g.:
FRF X

FRF also switches servo mode on for the axes of the motion platform of the hexapod (X, Y, Z, U, V, W).

FRF can be aborted by #24 (p. 146), STP (p. 252) and HLT (p. 193).

Use FRF? (p. 176) to check whether the reference move was successful.

FRF? (Get Referencing Result)

Description: Queries whether the specified axis is referenced or not.

Format: FRF? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller.

Response: {<AxisID>"="<uint> LF}

where

<uint> indicates whether the axis has been successfully referenced (=1) or not (=0).

Troubleshooting: Illegal axis identifier

Note: Axes are considered to be "referenced" when a reference move has been successfully completed with FRF (p. 174) or when the axes are equipped with absolute-measuring sensors.

FSA (Fast Scan with Automated Alignment)

Description: Starts a scanning procedure to determine the maximum intensity of an analog input signal in a plane. The search consists of two subprocedures:

"Coarse portion"; corresponds to the procedure that is started with the FSC command (p. 180)

"Fine portion"; corresponds to the procedure that is started with the AAP command (p. 147)

The fine portion is only executed when the coarse portion has successfully finished beforehand. For more detailed descriptions of the two subprocedures, see AAP and FSC.

Note:

For the fine portion, the scanning area is reset using the values for <Distance1> and <Distance2> specified with FSA, so that the start position of the fine portion is in the center of the specified area. As a result, the scanning area can be enlarged to up to double the original area.

➢ Make sure that the motion platform can safely move outside of the originally specified scanning area as well.

The plane in which the scanning area is located can be defined by one of the following axis pairs:

X Z

Y Z

X Y

FSA (and therefore the motion of the platform) has finished successfully when the fine portion has finished successfully, i.e., when three consecutive checks have confirmed that the local maximum is at the current position. The purpose of the checks is to reduce the influence of any superimposed noise signal.

FSA is finished unsuccessfully in the following cases:

- During the coarse portion, the intensity threshold has not been reached in the scanning area: the motion platform returns to the initial position.
- No gradient of the intensity was found during the fine portion: the motion platform returns to the start position of the fine portion.
- The specified area would be exceeded during the fine portion: The motion platform returns to the start position of the fine portion.
- #24 (p. 146), STP (p. 252) or HLT (p. 193) was sent: The motion platform remains at the current position.

FSS? (p. 187) can be used to check whether a scanning procedure has been successfully completed.

In order to check whether a scanning procedure is still going on, the motion status of the axes can be queried with #5 (p. 143).

Format: FSA <Axis1ID> <Distance1> <Axis2ID> <Distance2>
["L" <Threshold>] ["S" <ScanLineDistance>]
["SA" <StepSize>] ["A" <AnalogInputID>]

Arguments:	<Axis1ID>: Is one axis of the controller, format: string Axes X, Y, and Z are permissible. During the coarse portion, the platform moves in this axis from scanning line to scanning line by the distance given by <ScanlineDistance>. <Distance1>: Side length of the scanning area along the axis, format: double <Axis2ID>: Is one axis of the controller, format: string Axes X, Y, and Z are permissible. During the coarse portion, the scanning lines are located on this axis. <Distance2>: Side length of the scanning area along the axis, format: double "L": Required keyword for entering <Threshold> <Threshold>: Intensity threshold of the analog input signal, in V, format: Double "S": Required keyword for entering <ScanLineDistance> <ScanLineDistance>: Distance between the scanning lines, is only used during the coarse portion. Format: double "SA": Required keyword for entering <StepSize> <StepSize>: Starting value for the step size, is only used during the fine portion, format: double "A": Required keyword for entering <AnalogInputID> <AnalogInputID>: is the identifier of the analog input signal whose maximum intensity is sought, format: Integer
Response:	None

Notes:

The physical unit for specifying the <Distance1>, <Distance2>, <ScanLineDistance>, and <StepSize> can be queried with PUN? (p. 238).

The values for <Distance1> and <Distance2> must be identical.

The following default values are used when the corresponding arguments are omitted:
<Threshold>: 1.0 V
<ScanLineDistance>: 0.01 mm
<StepSize>: 0.001 mm
<AnalogInputID>: 1
These default values have only been checked for the H-810, H-811, and H-206 hexapod models. For other hexapod models, the ideal parameters must be determined experimentally.

The lower the system velocity is set with VLS (p. 262) during a scanning procedure, the greater the accuracy with which the maximum intensity is found.
Velocities in the range of under 1 mm/s are recommended.

If the values for distances are too large, the platform moves on an undefined path during a scanning procedure and it can tilt. Therefore, collisions are possible and it is also possible that the result of the scanning procedure is unsatisfactory. Measures for avoiding tilting:

- Select values of up to 0.2 mm for <Distance1> and <Distance2>.
- Align the motion platform suitably before starting the scanning procedure.
- Use suitable holders for the inputs and/or outputs of the optical element to be aligned on the motion platform so that the motion during the scanning procedure only takes place over short distances.

When the ***Trajectory Source*** parameter (ID 0x19001900) is set to 1, the dynamics profile must be defined by consecutive MOV commands, and FSA is not allowed.
For more information, see "Hexapod Motion" (p. 31).

Example:

FSA Y 0.2 Z 0.2 L 1 S 0.01 SA 0.001 A 2

Starts a scanning procedure in the YZ plane:

- Side length of the square area: 0.2 mm
- Intensity threshold for the coarse portion: 1 V
  When 1 V is reached, the motion platform stops, the scanning area is then reset, and the fine portion is started.

- Distance between the scanning lines (in the Z axis) for the coarse portion: 10 µm
- Start value of the step size for the fine portion: 1 µm
- Identifier of the analog input channel whose maximum intensity is sought: 2

FSC (Fast Scan with Abort)

Description:

Starts a scanning procedure which scans a specified area ("scanning area") until the analog input signal reaches a specified intensity threshold.

The scanning procedure started with FSC corresponds to the "coarse portion" of the scanning procedure that is started with the FSA command (p. 176).

The plane in which the scanning area is located can be defined by one of the following axis pairs:
X Z
Y Z
X Y
U W
V W
U V

The scanning area is centered around the current position at the time the command was received ("initial position"). The size of the scanning area is determined by the values for <Distance1> and <Distance2>.

The scanning procedure starts in the corner of the scanning area where the following holds true:

- For <Axis1ID>:
  start position1 = initial position1 - <Distance1>/2
- For <Axis2ID>:
  start position2 = initial position2 - <Distance2>/2

If the intensity threshold has already been reached at the initial position, the start position will not be approached.

<Axis1ID> specifies the axis in which the platform moves from one scanning line to the next, <Axis2ID> specifies the axis in which the scanning lines are located. The distance between the scanning lines can be specified.

FSC (and therefore the motion of the platform) has finished successfully when the analog input signal reaches the specified intensity threshold. If necessary due to the braking distance, the platform will return to the position at which the intensity threshold was reached after stopping.

FSC is finished unsuccessfully in the following cases:

- The intensity threshold has not been reached in the scanning area: the motion platform returns to the initial position.
- #24 (p. 146), STP (p. 252) or HLT (p. 193) was sent: The motion platform remains at the current position.

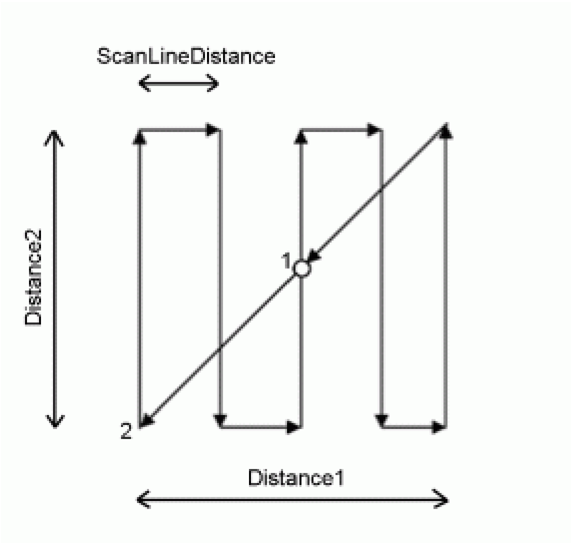


Tabelle 1: Motion sequence when the intensity threshold has not been reached in the scanning area

1 Initial position (= end position)
2 Starting position

FSS? (p. 187) can be used to check whether a scanning procedure has been successfully completed.

In order to check whether a scanning procedure is still going on, the motion status of the axes can be queried with #5 (p. 143).

A similar scanning procedure can be started with FSM (p. 184). In contrast to FSC, FSM scans the entire scanning area so that the global maximum is found if there are several local maximums.

Format: FSC <Axis1ID> <Distance1> <Axis2ID> <Distance2>
["L" <Threshold>] ["S" <ScanLineDistance>] ["A" <AnalogInputID>]

Arguments:	<Axis1ID>: Is the axis in which the platform moves from scanning line to scanning line by the distance given by <ScanlineDistance>. Format: String Axes X, Y, Z, U, V, and W are permissible. <Distance1>: Side length of the scanning area along the axis, format: double <Axis2ID>: Is the axis in which the scanning lines are located, format: string Axes X, Y, Z, U, V, and W are permissible. <Distance2>: Side length of the scanning area along the axis, format: double "L": Required keyword for entering <Threshold> <Threshold>: Intensity threshold of the analog input signal, in V, format: Double "S": Required keyword for entering <ScanLineDistance> <ScanLineDistance>: Distance between two scanning lines. Format: double "A": Required keyword for entering <AnalogInputID> <AnalogInputID>: is the identifier of the analog input signal whose maximum intensity is sought, format: Integer
Response:	None
Notes:	The physical unit for specifying the <Distance1>, <Distance2>, and <ScanLineDistance> can be queried with PUN? (p. 238). The values for <Distance1> and <Distance2> must be identical. The following default values are used when the corresponding arguments are omitted: <Threshold>: 1.0 V <ScanLineDistance>: 0.01 <AnalogInputID>: 1 These default values have only been checked for the H-810, H-811, and H-206 hexapod models. For other hexapod models, the ideal parameters must be determined experimentally.

The lower the system velocity is set with VLS (p. 262) during a scanning procedure, the greater the accuracy with which the threshold of the intensity is found.

Velocities in the range of under 1 mm/s are recommended.

If the values for distances or angles are too large, the platform moves on an undefined path during a scanning procedure and it tilt. Therefore, collisions are possible and it is also possible that the result of the scanning procedure is unsatisfactory. Measures for avoiding tilting:

- Select suitable values for <Distance1> and <Distance2>. For the hexapod models H-810, H-811, and H-206, 0.2 mm or 0.2 degrees should not be exceeded; for other hexapod models as well as in the case of changed settings for coordinate systems and the pivot point, the ideal values have to be determined experimentally.
- Set the velocity for the motion platform of the hexapod as low as possible (with the VLS command).
- Align the motion platform suitably before starting the scanning procedure.
- Use suitable holders for the inputs and/or outputs of the optical element to be aligned on the motion platform so that the motion during the scanning procedure only takes place over short distances or angles.

When the ***Trajectory Source*** parameter (ID 0x19001900) is set to 1, the dynamics profile must be defined by consecutive MOV commands, and FSC is not allowed.

For more information, see "Hexapod Motion" (p. 31).

Example: FSC Y 0.2 Z 0.2 L 1 S 0.05 A 2

Starts a scanning procedure in the YZ plane:

- Side length of the square area: 0.2 mm
- Intensity threshold: 1 V
- Distance between the scanning lines (in the Z axis): 50 µm
- Identifier of the analog input channel whose maximum intensity is sought: 2

FSM (Fast Scan to Maximum)

Description:

Starts a scanning procedure to determine the global maximum intensity of an analog input signal in a plane.

Completely scans a specified area ("scanning area") for the intensity of the analog input signal. In the case of several maximum intensities in the scanning area, this prevents only a local maximum from being found instead of the global maximum.

The plane in which the scanning area is located can be defined by one of the following axis pairs:

X Z
Y Z
X Y
U W
V W
U V

The scanning area is centered around the current position at the time the command was received ("initial position"). The size of the scanning area is determined by the values for <Distance1> and <Distance2>.

The scanning procedure starts in the corner of the scanning area where the following holds true:

- For <Axis1ID>:
  start position1 = initial position1 - <Distance1>/2
- For <Axis2ID>:
  start position2 = initial position2 - <Distance2>/2

<Axis1ID> specifies the axis in which the platform moves from one scanning line to the next, <Axis2ID> specifies the axis in which the scanning lines are located. The distance between the scanning lines can be specified.

FSM (and therefore the motion of the platform) has finished successfully when the two following conditions are met:

- The analog input signal has reached the specified intensity threshold in the scanning area at least once.
- The motion platform has returned from the end position of the scanning area to the position with maximum intensity.

FSM is finished unsuccessfully in the following cases:

- The intensity threshold has not been reached in the scanning area: the motion platform returns to the initial position.
- #24 (p. 146), STP (p. 252) or HLT (p. 193) was sent: The motion platform remains at the current position.

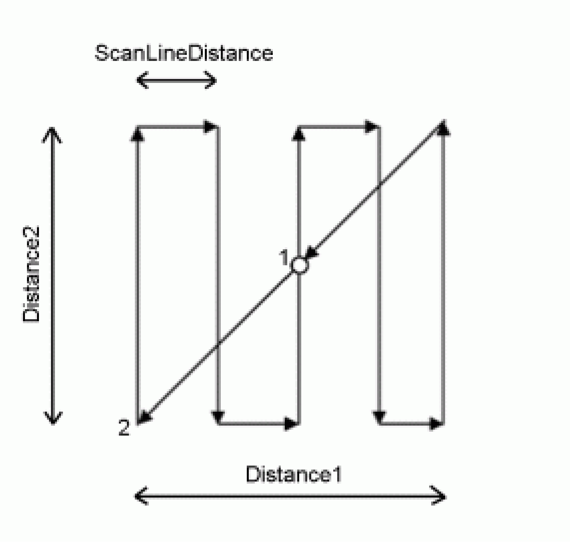


Tabelle 2: Motion sequence when the intensity threshold has not been reached in the scanning area

1 Initial position (= end position)
2 Starting position

FSS? (p. 187) can be used to check whether a scanning procedure has been successfully completed.

In order to check whether a scanning procedure is still going on, the motion status of the axes can be queried with #5 (p. 143).

A similar scanning procedure can be started with FSC (p. 180). In contrast to FSM, FSC has already finished successfully when the intensity threshold is reached for the first time.

Format: FSM <Axis1ID> <Distance1> <Axis2ID> <Distance2>
["L" <Threshold>] ["S" <ScanLineDistance>] ["A" <AnalogInputID>]

Arguments:	<Axis1ID>: Is the axis in which the platform moves from scanning line to scanning line by the distance given by <ScanlineDistance>. Format: String Axes X, Y, Z, U, V, and W are permissible. <Distance1>: Side length of the scanning area along the axis, format: double <Axis2ID>: Is the axis in which the scanning lines are located, format: string Axes X, Y, Z, U, V, and W are permissible. <Distance2>: Side length of the scanning area along the axis, format: double "L": Required keyword for entering <Threshold> <Threshold>: Intensity threshold of the analog input signal, in V, format: Double "S": Required keyword for entering <ScanLineDistance> <ScanLineDistance>: Distance between two scanning lines, format: double "A": Required keyword for entering <AnalogInputID> <AnalogInputID>: is the identifier of the analog input signal whose maximum intensity is sought, format: Integer
Response:	None
Notes:	The physical unit for specifying the <Distance1>, <Distance2>, and <ScanLineDistance> can be queried with PUN? (p. 238). The values for <Distance1> and <Distance2> must be identical. The following default values are used when the corresponding arguments are omitted: <Threshold>: 1.0 V <ScanLineDistance>: 0.01 <AnalogInputID>: 1 These default values have only been checked for the H-810, H-811, and H-206 hexapod models. For other hexapod models, the ideal parameters must be determined experimentally. In the case of intensity distributions with secondary maxima, FSM with a subsequent AAP (p. 147) is to be preferred over FSC (p. 180) or FSA (p. 176).

The lower the system velocity is set with VLS (p. 262) during a scanning procedure, the greater the accuracy with which the maximum intensity is found.
Velocities in the range of under 1 mm/s are recommended.

If the values for distances or angles are too large, the platform moves on an undefined path during a scanning procedure and it tilt. Therefore, collisions are possible and it is also possible that the result of the scanning procedure is unsatisfactory. Measures for avoiding tilting:

- Select suitable values for <Distance1> and <Distance2>. For the hexapod models H-810, H-811, and H-206, 0.2 mm or 0.2 degrees should not be exceeded; for other hexapod models as well as in the case of changed settings for coordinate systems and the pivot point, the ideal values have to be determined experimentally.
- Set the velocity for the motion platform of the hexapod as low as possible (with the VLS command).
- Align the motion platform suitably before starting the scanning procedure.
- Use suitable holders for the inputs and/or outputs of the optical element to be aligned on the motion platform so that the motion during the scanning procedure only takes place over short distances or angles.

When the ***Trajectory Source*** parameter (ID 0x19001900) is set to 1, the dynamics profile must be defined by consecutive MOV commands, and FSM is not allowed.
For more information, see "Hexapod Motion" (p. 31).

Example: FSM Y 0.2 Z 0.2 L 1 S 0.05 A 2

Starts a scanning procedure in the YZ plane:

- Side length of the square area: 0.2 mm
- Intensity threshold: 1 V
- Distance between the scanning lines (in the Z axis): 50 µm
- Identifier of the analog input channel whose maximum intensity is sought: 2

FSS? (Get Status of Fast Scan Routines)

Description: Gets the status of the last scanning procedure that was started.

In order to check whether a scanning procedure is still going on, the motion status of the axes can be queried with #5 (p. 143).

Format:	FSS?
Arguments:	None
Response:	<uint> indicates the status of the last scanning procedure that was started. 1: Scanning procedure has been successfully completed 0: Scanning procedure is still going on or has been unsuccessfully completed. For details on successful and unsuccessful completion, see the descriptions of the scanning procedures.
Note:	FSS? gets the status of scanning procedures that are started with the following commands: AAP (p. 147), FIO (p. 165), FLM (p. 168), FLS (p. 171), FSA (p. 176), FSC (p. 180), FSM (p. 184)
Example:	Send: AAP Y 0.1 Z 0.1 SA 0.001 N 3 A 2 Send: FSS? Receive: 0 Note: The scanning procedure is still running or has not been successfully completed. Send: FSS? Receive: 1 Note: The scanning procedure started with AAP has been successfully completed, i.e., a determined number of checks (number ≠ zero) has confirmed that the local maximum intensity is present at the current position.

GWD? (Get Wave Table Data)

Description:	Queries waveform for specified wave table.
Format:	GWD? [<StartPoint> <NumberOfPoints> [{<WaveTableID>}]]
Arguments:	<StartPoint> is the start point in the wave table, begins with index 1 <NumberOfPoints> is the number of points to be read per table <WaveTableID> is one wave table of the controller
Response:	The wave table contents (waveform) in GCS array format (refer to the separate manual for the GCS array, SM 146E, and the example below)

Notes: Depending on the waveform definition with WAV (p. 265), the wave tables can have different lengths.

- ➢ Only get wave tables of the same length in the same command.
- ➢ Get the contents of wave tables of different lengths with separate commands.

If wave tables of different lengths are queried together, the response will contain a maximum of as many points as the shortest wave table or the query will produce an error.

HDI? (Get Help For Interpretation Of DIA? Response)

Description: Shows descriptions and physical units for the measurands that can be queried with the DIA? command (p. 154).

Format: HDI?

Arguments: None

Response {<MeasureID>"="<Description>TAB<PhysUnit> LF}

where

<MeasureID> is the identifier of the measurand

<Description> is the name of the measurand

<PhysUnit> is the physical unit of the measurand.

HDR? (Get All Data Recorder Options)

Description: Lists a help string which contains all information available on data recording (record options and trigger options, information on additional parameters and commands concerning data recording).

Format: HDR?

Arguments: None

Response #RecordOptions
{<RecOption>"="<DescriptionString>[of <Channel>]}

#TriggerOptions
[{<TriggerOption>"="<DescriptionString>}]

#Parameters to be set with SPA
[{<ParameterID>"="<DescriptionString>}]

#Additional information
[{<Command description>"("<Command>")"}]

#Sources for Record Options
[{<RecOption>"="<Source>}]

end of help

Note: The HDR response lists all the possible record options for the C-887.

Example:

```
hdr?
#RecordOptions
0=Nothing is recorded
1=Commanded position of axis
2=Real position of axis
3=Position error of axis
8=Measurement time
17=Input value of channel, calculated in Volt
18=Input value of channel, directly from
channel, without dimension
150=Input value of channel, calculated in Volt,
input calculated made with SIC
70=Commanded velocity of axis
71=Commanded acceleration of axis
73=Output of axis
74=Current proportion to the current error of
axis
75=Current integrated position error of axis
76=Current derivative position error of axis
80=Status register of axis
86=Fifo Size for continuous position mode
84=Real position of second sensor in axis
87=Commanded position for continuous position
mode
#TriggerOptions
0=No trigger. Exception: STE and IMP always
trigger data recorder
1=Trigger with next command that changes the
position, default setting
2=Trigger with next command, resets trigger
settings to 0
4=Trigger immediately, resets trigger settings
to 0
6=Trigger with next command that changes the
position, resets trigger settings to 0
#Parameters to be set with SPA
```

```
0x16000000=Data Recorder   Data Recorder Table
Rate
0x16000201=Data Recorder   Data Recorder Points
Per Table
#Additional information
Set data recorder configuration with DRC
Get data recorder configuration with DRC?
Get number of recorded points with DRL?
Get recorded data values with DRR?
Set data recorder trigger source with DRT
Get data recorder trigger source with DRT?
Set record table rate (in cycles) with RTR
Get record table rate (in cycles) with RTR?
Tell number of data recorder tables with TNR?
#Sources for Record Options
0 = X Y Z U V W 1 2 3 4 5 6
1 = X Y Z U V W 1 2 3 4 5 6
2 = X Y Z U V W 1 2 3 4 5 6
3 = 1 2 3 4 5 6
8 = X Y Z U V W 1 2 3 4 5 6
17 = 1 2 3 4
18 = 1 2 3 4
150 = 1 2 3 4
70 = 1 2 3 4 5 6
71 = 1 2 3 4 5 6
73 = 1 2 3 4 5 6
74 = 1 2 3 4 5 6
76 = 1 2 3 4 5 6
75 = 1 2 3 4 5 6
80 = 1 2 3 4 5 6
84 = 1 2 3 4 5 6
87 = 1 2 3 4 5 6
end of help
```

HIB? (Get State Of HID Button)

Description: Queries the current state of the specified button of the specified HID.

Format: HIB? [{<HIDeviceID> <HIDeviceButton>}]

Arguments:	<HIDeviceID> is an HID connected to the controller; see below for more details. <HIDeviceButton> is an HID button; see below for more details.
Response:	{<HIDeviceID> <HIDeviceButton> "="<HIDButtonState>} where <HIDButtonState> specifies the status of the button as integer value: The possible values depend on the button type. The value range for the individual buttons can be queried with the HIS? command. When only the values 0 and 1 are permitted, they have the following meaning: 0 = Button not pressed, 1 = Button pressed The meaning of values > 1 depends on the HID.
Notes:	The C-887 supports one human interface device (identifier: 1) and two of its buttons (identifiers: 1 to 2). The human interface device can be connected to either of the USB interfaces of the C-887 . If a C-887.MC2 or a C-887.MC control unit is connected, the status of both pushbuttons can be queried with the HIB? command. The value of the ***HID Device Button Mode*** parameter (ID 0x0E001600) specifies the behavior of the pushbuttons: ▪ The parameter value is 0 (default): The pushbuttons trigger actions (stop, reference move) according to the descriptions in the documentation for the control unit. ▪ The parameter value is 1: The pushbuttons do not trigger any actions. The value of the parameter can be changed with the SPA command (p. 244).

HLP? (Get List Of Available Commands)

Description:	Lists a help string which contains all commands available.
Format:	HLP?
Arguments:	none
Response:	List of commands available
Troubleshooting:	Communication breakdown
Note:	The HLP? response contains the commands provided by the current command level. See CCL (p. 151) for more information.

HLT (Halt Motion Smoothly)

Description:	Stops the motion of specified axes smoothly. See the notes below for further details. Error code 10 is set. #24 (p. 146) and STP (p. 252) in contrast abort current motion as fast as possible for the controller without taking care of maximum velocity and acceleration.
Format:	HLT [{<AxisID>}]
Arguments:	<AxisID>: is one axis of the controller, if left out, all axes are stopped
Response:	none
Troubleshooting:	Illegal axis identifier
Notes:	HLT stops all axis motion caused by motion commands, fast alignment routines, scanning procedures or wave generator output, and stops the reference move. After the axes are stopped, their target positions are set to their current positions. HLT does **not** stop macros.

HPA? (Get List Of Available Parameters)

Description:	Responds with a help string that contains all available parameters with short descriptions. Refer to "Parameter Overview" (p. 302) for further information.
Format:	HPA?
Arguments:	None
Response	{<PamID>"="<string> LF} where <PamID> is the ID of one parameter, hexadecimal format <string> is a string which describes the corresponding parameter.

Notes: The string has the following format:

<CmdLevel>TAB<MaxItem>TAB<DataType>TAB<FunctionGroupDescription>TAB<ParameterDescription>[{TAB<PossibleValue>"="<ValueDescription>}]

where

<CmdLevel> is the command level that allows write access to the parameter value

<MaxItem> is the maximum number of elements of the same type that the parameter affects. (The meaning of "element" depends on the parameter; this can refer to an axis, a hexapod strut, an input signal channel, or the entire system.)

<DataType> is the data type of the parameter value, it can be INT, FLOAT or CHAR

<FunctionGroupDescription> is the name of the function group which the parameter belongs to. (Parameters are grouped according to their purpose in order to clarify their relation to each other.)

<ParameterDescription> is the parameter name

<PossibleValue> is a value from the permissible data range

<ValueDescription> is the meaning of the corresponding value

The parameter listing varies depending on the optional accessories that are installed. Furthermore, the list can change at the runtime of the C-887 if the assignment of positioner types to axes A and B is changed, see also "Operating Parameters for Axes A and B" (p. 30).

For querying and changing parameter values, see "Adapting Settings" (p. 299).

IFC? (Get Current Interface Parameters)

Description: Queries the values of the interface parameters for communication from volatile memory.

Format: IFC? [{<InterfacePam>}]

Arguments: <InterfacePam> is the interface parameter to be queried, see below for possible values.

Response: {<InterfacePam>"="<PamValue> LF}

where

<PamValue> gives the value of the interface parameter from volatile memory.

<InterfacePam> can be RSPORT, RSBAUD, RSHSHK, IPADR, IPSTART, IPMASK, IPMAXCONN, MACADR or TERMSTR.

For <InterfacePam> = RSPORT, <PamValue> returns the port used for RS-232 communication:
1

For <InterfacePam> = RSBAUD, <PamValue> indicates the baud rate used for RS-232 communication.

For <InterfacePam> = RSHSHK, <PamValue> returns the handshake setting for RS-232 communication:
1 = RTS/CTS

For <InterfacePam> = IPSTART, <PamValue> indicates the current setting of the startup behavior for the configuration of the IP address for TCP/IP communication:
0 = The IP address defined with IPADR is used.
1 = DHCP is used to obtain the IP address

For <InterfacePam> = IPADR, the first four parts of <PamValue> indicate the IP address which is currently used for TCP/IP communication; the last part indicates the port.

For <InterfacePam> = IPMASK, <PamValue> indicates the IP mask setting that is currently used for TCP/IP communication, in the format uint.uint.uint.uint.

For <InterfacePam> = IPMAXCONN, <PamValue> indicates the maximum permissible number of connections for TCP/IP communication.

For <InterfacePam> = MACADR, the <PamValue> indicates the unchangeable, unique address of the network hardware in the C-887.

For <InterfacePam> = TERMSTR, <PamValue> returns the termination character for the commands of the GCS:
0 = LineFeed (ASCII character 10)

IFS (Set Interface Parameters as Default Values

Description:	Stores interface parameters. The default parameters for the interface are changed in the nonvolatile memory, but the currently active parameters are not. Settings made with IFS become active after the next power-on or reboot.
Format:	IFS <Pswd> {<InterfacePam> <PamValue>}
Arguments:	<Pswd> is the password for writing to the nonvolatile memory, default is "100" <InterfacePam> is the interface parameter to be changed, see below <PamValue> specifies the value of the interface parameter, see below
Response:	None
	The following interface parameters can be set: RSBAUD <PamValue> indicates the baud rate to be used for RS-232 communication. Possible values are: 9600, 19200, 38400, 57600 and 115200. Default is 115200 IPADR The first four parts of <PamValue> specify the default IP address for TCP/IP communication, the last part specifies the default port to be used, default is 192.168.1.28:50000; Note: While the IP address can be changed, the port must always be 50000! IPSTART <PamValue> defines the startup behavior for configuration of the IP address for TCP/IP communication, 0 = The IP address defined with IPADR is used 1 = DHCP or Auto-IP is used to obtain IP address. (default); IPMASK <PamValue> indicates the subnet mask to be used for TCP/IP communication, in the form uint.uint.uint.uint, default is 255.255.255.0;
Response:	None

Notes:	**Note that the number of write cycles in the nonvolatile memory is limited. Write default settings only if necessary.** When IPSTART = 1, the C-887 uses the Auto-IP functionality for TCP/IP communication in a network where no DHCP server is present or directly with the PC. Using Auto-IP the network devices automatically configure their interfaces so that no manual setting of the IP addresses is necessary. Further interface parameters of the C-887 are write-protected: The default settings of these parameters can be queried with IFS? (p. 197). For more information, see "Establishing Communication via TCP/IP Interfaces" (p. 72)

IFS? (Get Interface Parameters as Default Values)

Description:	Queries the parameter values of the interface configuration stored in the nonvolatile memory (i.e. default settings)
Format:	IFS? [{<InterfacePam>}]
Arguments:	<InterfacePam> is the interface parameter to be queried. See below for possible values.
Response:	{<InterfacePam>"="<PamValue> LF} where <PamValue> is the value of the interface parameter in nonvolatile memory. <InterfacePam> can be RSPORT, RSBAUD, RSHSHK, IPADR, IPSTART, IPMASK, IPMAXCONN, MACADR and TERMSTR The following interface parameters are write-protected: For <InterfacePam> = RSPORT, <PamValue> returns the port used for RS-232 communication: 1 For <InterfacePam> = RSHSHK, <PamValue> returns the handshake setting for RS-232 communication: 1 = RTS/CTS For <InterfacePam> = IPMAXCONN, <PamValue> indicates the maximum permissible number of connections for TCP/IP communication.

For <InterfacePam> = MACADR, <PamValue> returns the unique address of the network hardware in the C-887.

For <InterfacePam> = TERMSTR, <PamValue> returns the termination character for the commands of the GCS:
0 = LineFeed (ASCII character 10)

For all other forms of <InterfacePam>, see IFS (p. 196).

IMP (Start Impulse and Response Measurement)

Description:	Starts an impulse and records the impulse response for the specified axis. The data recorder configuration, i.e., the assignment of data sources and record options to the recorder tables, can be set with DRC (p. 157). The recorded data can be read with the DRR? command (p. 160).
Format:	IMP <AxisID> <Amplitude>
Arguments:	<AxisID> is one axis of the controller <Amplitude> is the height of the impulse. See below for details.
Response:	None
Troubleshooting:	The target position resulting from the specified impulse height is out of limits.
Notes:	An "impulse" consists of a relative motion with the specified amplitude, followed by an equally large motion in the opposite direction. The impulse is executed relative to the current position. The pulse width of the impulse results from the value of the ***Pulse Length Factor*** parameter (ID 0x0E000900) multiplied by the axis-dependent cycle time (p. 348). The physical unit for specifying the <Amplitude> can be queried with PUN? (p. 238) The following is valid for the axes of the hexapod (X, Y, Z, U, V, W): ▪ Before the start of any motion, it is checked whether the motion platform can actually reach the commanded target position. You can query whether the target position can be reached with VMO? (p. 263) ▪ When the ***Trajectory Source*** parameter (ID 0x19001900) is set to 1, the dynamics profile must be defined by consecutive MOV commands. IMP is not permitted. For more information, see "Hexapod Motion" (p. 31).

JRC (Jump Relatively Depending On Condition)

Description:	Jumps relatively depending on a specified condition of the following type: one specified value is compared with a queried value according to a specified rule. Can only be used in macros.
Format:	JRC <Jump> <CMD?> <OP> <Value>
Arguments:	<Jump> is the size of the relative jump. -1 means the macro execution pointer jumps back to the previous line, 0 means the command is executed again, which is the same behavior as with WAC (p. 264). 1 jumps to the next line, making the command unnecessary, and 2 jumps over the next command. Only jumps within the current macro are allowed. <CMD?> is one query command in its usual notation. The response has to be a single value and not more. For an example see below. <OP> is the operator to be used. The following operators are possible: = <= < > >= != Important: There must be a blank space before and after the operator! <Value> is the value to be compared with the response to <CMD?>.
Response:	None
Troubleshooting:	Check proper jump target

KCP (Copy Coordinate System)

Description:	Generates a copy of a coordinate system.
Format:	KCP <CSNameSource> <CSNameCopy>
Arguments:	<CSNameSource> is the name of the coordinate system of which a copy is to be generated. PI_Base, PI_Levelling, ZERO and HEXAPOD cannot be copied. <CSNameCopy> is the name of the copy of the coordinate system.
Response:	None
Notes:	PI_Base, PI_Levelling, ZERO and HEXAPOD cannot be copied. Options for creating the copy: ▪ <CSNameCopy> is a new name. The copy is created as a new coordinate system with this name.

- <CSNameCopy> is the name of an existing coordinate system that is not in use. This overwrites the coordinate system.

The link to the parent in a chain of coordinate systems is copied. The link to parents is not copied.

The copy is generated in the volatile memory. `WPA SKS` (p. 276) can be used to write the copy to the nonvolatile memory.

KEN (Activate Coordinate System)

Description: Activates the specified coordinate system. The scope of the settings that are influenced by activation depends on the coordinate system type, see below.

Position values for the hexapod's motion platform (get with POS? (p. 238)) refer to the active operating coordinate system.

When applying the work-and-tool concept:

- The work-and-tool concept uses a combination of two active operating coordinate systems. Generally, the combination consists of one active KST type and one KSW type coordinate system. If a coordinate system is active for only one of the two types, a substitute is automatically used for the other type.
- The current position of the hexapod's motion platform queried with POS? can be considered the position of the tool coordinate system in the work coordinate system.

Enabling coordinate systems using KEN does not initiate motion.

Format: KEN <CSName>

Arguments: <CSName> is the name of the coordinate system to be activated.

Response: None

Notes: Before activating a coordinate system, KEN checks that the definition of this coordinate system and its link are correct. If the coordinate system is not correctly defined, it is not activated.

KEN can be used to activate coordinate systems of the following types: KSD, KSF, KSW, KST, KSB(USER), KSB(PI), KLF(USER), KLF(PI), KLD(USER), KLD(PI), ZERO

Exactly one coordinate system or exactly one coordinate system combination is active from the following groups of coordinate systems:

- Operating coordinate system: A ZERO or KSF or KSD type coordinate system or - for the work-and-tool concept - a

combination of KSW/KST, or ZERO/KST, or KSW/ZERO type coordinate
- Leveling coordinate system (KLD(USER), or KLD(PI), or KLF(USER), or KLF(PI) type)

By activating a coordinate system for one of the groups, the coordinate system or the coordinate system combination that was previously activated for this group is deactivated simultaneously.

Sending KEN ZERO reactivates the ZERO operating coordinate system that is activated by default. When command level 1 is active (see CCL (p. 151)), KEN ZERO also reactivates the PI_Levelling leveling coordinate system, which was active by default, but not the PI_Base orientational coordinate system, which was activated by default (this can be reactivated by sending KEN PI_Base). When `DPA SKS` (p. 156) is sent, all coordinate systems activated by default can be reactivated independent of the currently active command level.

The active operating coordinate system specifies values for the following settings (for the work-and-tool concept, the values are specified using the combination of two operating coordinate systems):
- NLM (p. 234): Limit for the low end of the axis travel range
- PLM (p. 236): Limit for the high end of the axis travel range
- SSL (p. 248): Activation state of the soft limits of the axis
- SPI (p. 245): Coordinates of the pivot point (only for KSF and ZERO type coordinate systems)
- If supported by the controller:
  SST (p. 249): Step size for motion initiated by a manual control unit

Before activating leveling and orientational coordinate systems (KLD(), KLF() and KSB() types), it is necessary to switch to command level 1 (see CCL).

The coordinate systems are activated and deactivated in the volatile memory. `WPA SKS` (p. 276) can be used to write the activation state to the nonvolatile memory.

KEN? (Get Active Coordinate System)

Description:	Lists the names of the active coordinate systems and displays their type.
Format:	KEN? [{<CSName>}]
Arguments:	<CSName> is the name of an active coordinate system. Illegal: ZERO. When <CSName> is omitted, all active coordinate systems are listed.

Response: {<CSName>"="<CSType>}

where

<CSType> indicates the coordinate system type.

Notes: KEN? queries the volatile memory.

When the ZERO operating coordinate system is active, it is **not** displayed in the response to KEN? and the response contains only the following:

- The active leveling coordinate system, i.e., a KLD(PI), or KLD(USER), or KLF(PI), or KLF(USER) type coordinate system
- The enabled orientational coordinate system, i.e., a KSB(PI) or KSB(USER) type coordinate system

If <CSName> is given in the query and the corresponding coordinate system is not enabled, the C-887 sends an empty response and sets an error (get error code using ERR? (p. 164)).

KET? (Get Active Coordinate System Types)

Description: Lists the active coordinate system types and displays the names of the corresponding coordinate systems.

Format: KET? [{<CSType>}]

Arguments: <CSType> is an active coordinate system type. Possible values: KSW, KST, KSF, KSD, KLD(PI), KLD(USER), KLF(PI), KLF(USER), KSB(PI), KSB(USER)

When <CSType> is omitted, all active coordinate system types are listed.

Response: {<CSType>"="<CSName>}

where

<CSName> indicates the name of the coordinate system.

Notes: KET? queries the volatile memory.

When the ZERO type operating coordinate system is enabled, this type is **not** displayed in the response to KET?, and the response contains only the following:

- The type of the enabled leveling coordinate system, i.e., KLD(PI) or KLD(USER) or KLF(PI) or KLF(USER)
- The type of the active orientational coordinate system, i.e., KSB(PI) or KSB(USER)

If <CSType> is given in the query and no coordinate system of the corresponding type is enabled, the C-887 sends an empty response and sets an error (get the error code using ERR? (p. 164)).

KLC? (Get Properties Of Work-And-Tool Combinations)

Description: Lists the properties of the coordinate system combinations for the work-and-tool concept in the volatile memory.

The work-and-tool concept uses a combination of two active operating coordinate systems. Generally, the combination consists of one KST type active coordinate system and one of the KSW type. If a coordinate system is active for only one of the two types, a substitute is automatically used for the other type. Coordinate systems used as a substitute are listed under the name "Zero" in the response to KLC?

A combination is created in the volatile memory when a KST or KSW type coordinate system is activated. The combinations remain in the volatile memory even if the KST or KSW type coordinate systems contained therein are no longer active.

When a KST or KSW type coordinate system is deleted with KRM (p. 214), the response to KLC? no longer lists the combinations in which this coordinate system was involved.

`WPA SKS` (p. 276) can be used to write the combinations in the volatile memory to the nonvolatile memory.

KLS? (p. 211) can be used to get the properties of the coordinate systems in the volatile memory.

Format: KLC? [<CSName1>[<CSName2>[<Item1>[<Item2>]]]]

Arguments: <CSName1> is the name of a KST or KSW type coordinate system that is part of a combination in the volatile memory.

<CSName2> is the name of a KST or KSW type coordinate system that is part of a combination in the volatile memory.

<Item1> is a property of the axes of the controller for the queried combination of coordinate systems. Possible values:

- NLM: Limit for the low end of the axis travel range (“soft limit”)
- PLM: Limit for the high end of the axis travel range (“soft limit”)
- SSL: Activation state of the soft limits of the axis
- If supported by the controller:
  SST: Step size for motion initiated by a manual control unit

The settings for the properties of the currently active combination can be changed with the corresponding commands and saved with WPA.

<Item2> is an axis of the controller, possible values: X, Y, Z, U, V, W

If the properties of all coordinate system combinations are to be listed, all arguments are omitted.

Response: <String>

<String> contains information in XML format on the combinations of coordinate systems in the volatile memory.

The response structure depends on the number of arguments in the command sent. Possible responses, depending on the number of arguments in the example of coordinate systems Node1, Node2, Node3 and Node4:

Sent

```
KLC?
```

Response:

```
<CombinedCoordinateSystem>[SP][LF]
[TAB] <NODE1.NODE2 Name="NODE1.NODE2"
Work="NODE1" Tool="NODE2">[SP][LF]
[TAB] [TAB] <NLM X="-3.0" … W="-5.0"/>[SP][LF]
[TAB] [TAB] <PLM X="3.0" … W="5.0"/>[SP][LF]
[TAB] [TAB] <SSL X="1" Y="0" … W="1"/> [SP][LF]
[TAB] [TAB] <SST X="0.1" … W="0.2"/>[SP][LF]
[TAB] </NODE1.NODE2>[SP][LF]
[TAB] [TAB] <NODE1.NODE4  Name="NODE1.NODE4"
Work="NODE1" Tool="NODE4"> [SP][LF]
[TAB] [TAB] <NLM X="-4.0" … W="-3.4"/>[SP][LF]
[TAB] [TAB] <PLM X="2.0" … W="2.0"/>[SP][LF]
[TAB] [TAB] <SSL X="1" Y="0" … W="1"/> [SP][LF]
[TAB] [TAB] <SST X="0.2" … W="0.15"/>[SP][LF]
[TAB] </NODE1.NODE4>[SP][LF]
…
</CombinedCoordinateSystem>[LF]
```

Sent

```
KLC? Node1
```

Response:

```
<CombinedCoordinateSystem>[SP][LF]
[TAB] <NODE1.NODE2 Name="NODE1.NODE2"
Work="NODE1" Tool="NODE2"> [SP][LF]
[TAB] [TAB] <NLM X="-3.0" … W="-5.0"/>[SP][LF]
```

```
[TAB] [TAB] <PLM X="3.0"  … W="5.0"/>[SP][LF]
[TAB] [TAB] <SSL X="1" Y="0" … W="1"/> [SP][LF]
[TAB] [TAB] <SST X="0.1" … W="0.2"/>[SP][LF]
[TAB] </NODE1.NODE2 >[SP][LF]
[TAB] <NODE1.NODE4 Name="NODE1.NODE4" 
Work="NODE1" Tool="NODE4"> [SP][LF]
[TAB] [TAB] <NLM X="-1.0" … W="-7.0"/>[SP][LF]
[TAB] [TAB] <PLM X="1.1"  … W="10.0"/>[SP][LF]
[TAB] [TAB] <SSL X="1" Y="0" … W="1"/> [SP][LF]
[TAB] [TAB] <SST X="0.2" … W="0.21"/>[SP][LF]
[TAB] </NODE1.NODE4 >[SP][LF]
…
</CombinedCoordinateSystem>[LF]
```

Sent

```
KLC? Node1 Node2
```

Response:

```
<CombinedCoordinateSystem>[SP][LF]
[TAB] <NODE1.NODE2 Name="NODE1.NODE2" 
Work="NODE1" Tool="NODE2"> [SP][LF]
[TAB] [TAB] <NLM X="-3.0" … W="-5.0"/>[SP][LF]
[TAB] [TAB] <PLM X="3.0" …  W="5.0"/>[SP][LF]
[TAB] [TAB] <<SSL X="1" Y="0" … W="1"/> [SP][LF]
[TAB] [TAB] <SST X="0.1" … W="0.2"/>[SP][LF]
[TAB] </NODE1.NODE2 >[SP][LF]
</CombinedCoordinateSystem>[LF]
```

Sent

```
KLC? Node1 Node2 PLM
```

Response:

```
<PLM X="3.0" … W="5.0"/>[LF]
```

Sent

```
KLC? Node1 Node2 PLM X
```

Response:

```
X = 3.0 [LF]
```

KLD (Define Leveling Coordinate System By Specifying Values)

Description: Defines a KLD(USER) type leveling coordinate system for permanent correction of errors in the hexapod alignment (e.g., faulty mounting).

The leveling coordinate system is defined on the basis of measurements (e.g., using an interferometer) and corrects the linear displacement (X, Y, Z axes) and axis tilt (U, V, W axes) of the of the hexapod's motion platform.

If the linear displacement and axis tilt cannot be measured: Use KLF (p. 207) to define a leveling coordinate system.

The coordinate system is defined in the volatile memory. `WPA SKS` (p. 276) can be used to write the definition to the nonvolatile memory.

Format: KLD <CSName> [{<AxisID> <Offset>}]

Arguments: <CSName> is the name of the coordinate system to be defined.

<AxisID> is one axis of the controller. Possible values: X, Y, Z, U, V, W.

<Offset> is an offset which is added to the current position value of the axis after the reference move; in physical units.

For axes not given in the KLD command, the offset is set to zero.

Response: None

Notes: Before defining a leveling coordinate system, it is necessary to switch to command level 1 (see CCL (p. 151)).

Options for defining a coordinate system using KLD:

- <CSName> is a new name. The leveling coordinate system is created under this new name.
- <CSName> is the name of an existing coordinate system that is not in use. The coordinate system is overwritten by the definition created with KLD.

Recommended procedure for defining and activating a KLD(USER) type leveling coordinate system:

1. Do a reference move (see FRF (p. 174))
2. Measure the deviation between the position and orientation of the hexapod's motion platform and the position and orientation where X = 0, Y = 0, Z = 0, U = 0, V = 0, W = 0 is to be applicable in the future (measured by an external measuring instrument)
3. Switch to command level 1 by sending `CCL 1 advanced`
4. Define the leveling coordinate system using KLD by giving the measured discrepancies for the axes of the motion platform (offset values)
5. activate the leveling coordinate system (see KEN (p. 200))
6. Optional: Define the behavior after the reference move by setting the ***Behavior After Reference Move*** (ID 0x07030401) and ***Target***

For Motion After Reference Move (ID 0x07030402) parameters. In this way for example, the axes of the platform can be moved automatically to the zero position after the reference move.

- Value of parameter 0x07030401 = 0: The axis remains at the reference position after the reference move
- Value of parameter 0x07030401 = 1: After the reference move, the axis moves to the target position which is given by parameter 0x07030402

7. Save the settings by sending `WPA SKS`

The offset values which are displayed in the response to KLS? (p. 211) for the KLD(USER) type coordinate systems result via recalculation from all currently active coordinate systems. Therefore, they can change when active coordinate systems are changed. The offset values listed in the response to KLT? (p. 213), however, refer in each case to the specified parent in the chain and are therefore independent of the currently active coordinate systems.

The following applies for leveling coordinate systems defined with KLD:

- The leveling coordinate system is always the direct successor to the default PI_Levelling leveling coordinate system (automatic linking).
- The leveling coordinate system **cannot** be linked to other coordinate systems with KLN (p. 209)

DPA SKS (p. 156) reactivates the default PI_Levelling leveling coordinate system independently of the currently active command level; for details see KEN (p. 200).

KLF (Define Leveling Coordinate System At Current Position)

Description:

Defines a KLF(USER) type leveling coordinate system for permanent correction of errors in hexapod alignment (e.g., installation errors).

To define the leveling coordinate system, the motion platform after the reference move is commanded into the position and orientation for which X = 0, Y = 0, Z = 0, U = 0, V = 0, W = 0 is applicable in future. Sending KLF defines a coordinate system with offset values which are added, after the reference move, to the current position values for the axes; in physical units.

If the linear displacement (X, Y, Z axes) and axis tilt (U, V, W axes) are to be measured: Use KLD (p. 205) to define a leveling coordinate system.

The coordinate system is defined in the volatile memory. `WPA SKS` (p. 276) can be used to write the definition to the nonvolatile memory.

Format:	KLF <CSName>
Arguments:	<CSName> is the name of the coordinate system to be defined.
Response:	None

Notes:

Before defining a leveling coordinate system, it is necessary to switch to command level 1 (see CCL (p. 151)).

Options for defining a coordinate system with KLF:

- <CSName> is a new name. The leveling coordinate system is created under this new name.
- <CSName> is the name of an existing coordinate system that is not in use. The coordinate system is overwritten by the definition created with KLF.

Definition with KLF is only possible when the hexapod is not in motion.

Recommended procedure for defining and activating a KLF(USER) type leveling coordinate system:

1. Do a reference move (see FRF (p. 174))
2. Move to the position and orientation of hexapod's motion platform where X = 0, Y = 0, Z = 0, U = 0, V = 0, W = 0 is to be applicable in the future
3. Switch to command level 1 by sending `CCL 1 advanced`
4. Define the leveling coordinate system with KLF
5. Activate the leveling coordinate system (see KEN (p. 200))
6. Optional: Define the behavior after the reference move by setting the ***Behavior After Reference Move*** (ID 0x07030401) and ***Target For Motion After Reference Move*** (ID 0x07030402) parameters. In this way for example, the axes of the platform can be moved to the zero position automatically after the reference move.
   - Value of parameter 0x07030401 = 0: The axis remains at the reference position after the reference move
   - Value of parameter 0x07030401 = 1: After the reference move, the axis moves to the target position which is specified by parameter 0x07030402
7. Save the settings by sending `WPA SKS`

The offset values that are displayed in the response to KLS? (p. 211) for the KLF(USER) type coordinate systems are the result of recalculation from all currently active coordinate systems. Therefore, they can change when active coordinate systems are changed. The offset values listed in the response to KLT? (p. 213) however, refer in each case to the specified parent in the chain and are therefore independent of the currently active coordinate systems.

The following applies for leveling coordinate systems defined using KLF:

- The leveling coordinate system is always the direct successor to the default PI_Levelling leveling coordinate system (automatic linking).
- The leveling coordinate system **cannot** be linked to other coordinate systems with KLN (p. 209)

DPA SKS (p. 156) reactivates the default PI_Levelling leveling coordinate system independently of the currently active command level; for details see KEN (p. 200).

KLN (Link Coordinate Systems)

Description: Links two coordinate systems to create a chain of parent and child.

Format: KLN <ChildCS> <ParentCS>

Arguments: <ChildCS> is the name of the coordinate system that is to be attached in the chain as the child to <ParentCS>.

<ParentCS> is the name of the coordinate system that is to be the parent of <ChildCS> in the chain.

Response: None

Notes: Each coordinate system is part of at least one chain. For the basic structure of coordinate system chains, see "Coordinate Systems" (p. 38).

By default, the following coordinate systems are linked to form a chain:

- The HEXAPOD coordinate system, which is based on the configuration file with the geometric data of the hexapod, is the "origin" of all chains and the parent of the PI_Levelling leveling coordinate system (fixed linking)
- PI_Levelling is the parent of the PI_Base orientational coordinate system
- PI_Base is the parent of the ZERO operating coordinate system

The following applies for linked coordinate systems:

- The actual offset values for the position of the X, Y, Z, U, V, W axes result from the offset values for the parents linked to a coordinate system (for details, see KLT? (p. 213)).
- Each coordinate system has exactly one parent and can have one or more than one child.
- When a coordinate system is active, all parents in the chain are also in use and cannot be deleted or overwritten.

- When a coordinate system (unused) is deleted, its parent and child are linked to each another in the chain.

Limitations for creating chains using KLN:

- A coordinate system cannot be linked to itself.
- Although it is possible to form ring connections consisting of at least two coordinate systems, they cannot be activated with KEN (p. 200).
- KLN cannot attach a coordinate system in use as child of another coordinate system.
- KLN can be used to attach a coordinate system (not in use) as child of a coordinate system in use.
- Before linking orientational coordinate systems of the KSB(USER) type as successors, it is necessary to switch to command level 1 (see CCL (p. 151)).
- KLD(PI), KLF(PI), KLD(USER), KLF(USER) type coordinate systems and the HEXAPOD coordinate system cannot be linked with KLN.
- KLN cannot be used to attach the PI_Base coordinate system as child of another coordinate system.
- KLN can be used to link KSB(USER) type coordinate systems only to other KSB(USER) coordinate systems or as child of the PI_Base coordinate system.
- Using KLN, the ZERO coordinate system cannot be attached as successor to another coordinate system.

Coordinate systems are linked in the volatile memory with KLN. WPA SKS (p. 276) can be used to write the link to the nonvolatile memory.

KLN? (Get Coordinate System Chains)

Description:	Lists the components of the existing coordinate system chains. Each coordinate system is part of at least one chain. For the basic structure of coordinate system chains, see "Coordinate Systems" (p. 38).
Format:	KLN? [{<CSName>}]
Arguments:	<CSName> is the name of a coordinate system, the predecessors of which are to be listed in the chain. If the predecessors of all coordinate systems are to be listed, <CSName> is omitted.
Response:	{<CSName>"="<String>} where

<String> contains the names of the coordinate system predecessors in the chain. The "origin" of the chain is always given at the end of the line.

Notes: To ensure a clear overview, only the predecessors up to the ZERO coordinate system are listed for operating coordinate systems of the KSD, KSF, KSW and KST types. The predecessors of ZERO, however, can be specifically queried and are also contained as a separate line in the response if <CSName> is omitted from the query.

KLS? (Get Coordinate System Properties)

Description: Lists the properties of the coordinate systems that are present in the volatile memory.

`WPA SKS` (p. 276) can be used to write the coordinate systems in the volatile memory to the nonvolatile memory.

The properties of the coordinate system combinations for the work-and-tool concept that are present in the volatile memory can be queried using KLC? (p. 203)

Format: KLS? [<CSName>[<Item1>[<Item2>]]]

Arguments: <CSName> is the name of a coordinate system. Illegal: HEXAPOD.

<Item1> is a property of the axes of the controller. Possible values:

For all types of coordinate systems:

- POS: Offset for positions of the axes

Only for coordinate systems of the KSD, KSF and ZERO types:

- NLM: Limit for the low end of the axis travel range (“soft limit”)
- PLM: Limit for the high end of the axis travel range (“soft limit”)
- SSL: Activation state of the soft limits of the axis
- If supported by the controller:
  SST: Step size for motion initiated by a manual control unit

Only for coordinate systems of the KSF and ZERO types:

- SPI: Coordinates of the pivot point

<Item2> is an axis of the controller, possible values: X, Y, Z, U, V, W

If the properties of all coordinate systems are to be listed, all arguments are omitted.

Response: <String>

<String> contains information in XML format on the coordinate systems in the volatile memory.

The <String> structure depends on the number of arguments in the command sent.

If no argument or only the name of a coordinate system is given, <String> lists, for each coordinate system contained in the response, the data applicable for <Item1> and <Item2> as well as the information below:

- Name of the coordinate system (Name="")
- Name of the direct parent of the coordinate system in the chain (Parent="")
- Current use of the coordinate system;
  - Used="True": The coordinate system is in use, i.e., it is either active itself or it is a parent of the active coordinate system in its chain.
  - Used="False": The coordinate system is not used.
- Coordinate system type (Type="")

Note: The offset values which are displayed in the response to KLS? for the KLD(USER) and KLF(USER) type leveling coordinate systems are the result of recalculation from all currently active coordinate systems. Therefore, they can change when active coordinate systems are changed.

Example: The example below shows the response to KLS? for the coordinate systems existing by default.

```
KLS?
<SingleCoordinateSystem>
     <ZERO Name="ZERO" Parent="PI_BASE" Used="True" Type="ZERO">
          <POS X="0.000000" Y="0.000000" Z="0.000000"
U="0.000000" V="0.000000" W="0.000000"/>
          <NLM X="-10.000100" Y="-10.000100" Z="-10.000100"
U="-1.000100" V="-1.000100" W="-1.000100"/>
          <PLM X="10.000100" Y="10.000100" Z="10.000100"
U="1.000100" V="1.000100" W="1.000100"/>
          <SSL X="1" Y="1" Z="1" U="1" V="1" W="1"/>
          <SPI R="0.000000" S="0.000000" T="481.000000"/>
          <SST X="0.010000" Y="0.010000" Z="0.010000"
U="0.010000" V="0.010000" W="0.010000"/>
     </ZERO>
     <PI_BASE Name="PI_BASE" Parent="PI_LEVELLING" Used="True"
Type="KSB(PI)">
```

```
          <POS X="0.000000" Y="0.000000" Z="0.000000" U="0.000000" V="0.000000" W="0.000000"/>
     </PI_BASE>
     <PI_LEVELLING Name="PI_LEVELLING" Parent="HEXAPOD" Used="True" Type="KLD(PI)">
          <POS X="0.000000" Y="0.000000" Z="0.000000" U="0.000000" V="0.000000" W="0.000000"/>
     </PI_LEVELLING></SingleCoordinateSystem>
```

KLT? (Get Offset Resulting From A Chain)

Description: Lists, for a coordinate system, the actual offset values for the position of the X, Y, Z, U, V, W axes which result from the offset values of the predecessors linked to this coordinate system.

For operating coordinate systems, only the operating coordinate systems linked as predecessors up to the ZERO coordinate system are included in the calculation.

For orientational coordinate systems, only the orientational coordinate systems linked as predecessors up to the HEXAPOD coordinate system are included in the calculation.

For leveling coordinate systems, only the leveling coordinate systems linked as predecessors up to the HEXAPOD coordinate system are included in the calculation.

Format: KLT? [<StartCS> [<EndCS>]]

Arguments: <StartCS> is the name of the coordinate system for which the offset values resulting from its predecessors are to be queried.

<EndCS> is the name of a coordinate system linked as a predecessor of <StartCS>, which is to be used as the starting point for the offset calculation. If <EndCS> is omitted, the starting point for the calculation depends on the value for <StartCS>:

- <StartCS> is an operating coordinate system (KSD, KSF, KST, KSW, or ZERO type): Starting point is ZERO
- <StartCS> is an orientational or leveling coordinate system (KSB(PI), KSB(USER), KLD(PI), KLD(USER), KLF(PI), KLF(USER) types): Starting point is HEXAPOD

If the resultant offset values for all coordinate systems are to be listed, all arguments are omitted.

Response: <String>

<String> contains, for each queried coordinate system, a line with the following data:

- Name: The name of the coordinate system for which the resultant offset values are listed.
- EndCoordinateSystem: The name of the coordinate system which was used as the starting point for calculating the offset values.
- X, Y, Z, U, V, W: Resultant offset value for the corresponding axis.

Note: KLT? queries the volatile memory.

KLS? (p. 211) can be used to get the properties of the coordinate systems in the volatile memory.

Example: The example below shows the response to KLT? for the coordinate systems existing by default.

```
KLT?
Name=ZERO  EndCoordinateSystem=ZERO   X=0.000000 Y=0.000000
     Z=0.000000 U=0.000000 V=0.000000 W=0.000000
Name=PI_BASE    EndCoordinateSystem=HEXAPOD     X=0.000000
     Y=0.000000 Z=0.000000 U=0.000000 V=0.000000 W=0.000000
Name=PI_LEVELLING    EndCoordinateSystem=HEXAPOD     X=0.000000
     Y=0.000000 Z=0.000000 U=0.000000 V=0.000000 W=0.000000
```

KRM (Remove Coordinate System)

Description: Deletes a coordinate system.

Format: KRM <CSName>

Arguments: <CSName> is the name of the coordinate system to be deleted.

Response: None

Notes: A coordinate system in use (i.e., it is active itself or linked to the active coordinate system as a parent) cannot be deleted.

When a coordinate system (unused) is deleted, its parent and its child are linked to each another in the coordinate system chain.

Before deleting a KLD(USER), KLF(USER), and KSB(USER) type leveling coordinate system, it is necessary to switch to command level 1 (see CCL (p. 151)).

The coordinate system is deleted from the volatile memory. `WPA SKS` (p. 276) can be used to transfer the deletion to the nonvolatile memory.

KSB (Define Orientational Coordinate System)

Description: Defines a KSB(USER) type orientational coordinate system for a permanent change of direction of the X, and/or Y, and/or Z axes.

The direction of the axes is changed by rotating the coordinate system in 90° increments as follows:

- Rotation around X, i.e., angular value specified for U, changes the direction of the Y and Z axes
- Rotation around Y, i.e., angular value specified for V, changes the direction of the X and Z axes
- Rotation around Z, i.e., angular value specified for W, changes the direction of the X and Y axes

The coordinate system is defined in the volatile memory. `WPA SKS` (p. 276) can be used to write the definition to the nonvolatile memory.

Format: KSB <CSName> [{<AxisID> <Angle>}]

Arguments: <CSName> is the name of the coordinate system to be defined.

<AxisID> is one axis of the controller. Possible values: U, V, W.

<Angle> is the angle around which the axis is to be rotated. Possible values: 0, 90, 180, 270, -90, -180, -270 (unit: Degree).

For axes not given in the KSB command, the angle is set to zero.

Response: None

Notes: Before defining an orientational coordinate system, it is necessary to switch to command level 1 (see CCL (p. 151)).

Options for defining a coordinate system using KSB:

- <CSName> is a new name. The new orientational coordinate system is created under this name and the active orientational coordinate system becomes its parent automatically
- <CSName> is the name of an existing coordinate system that is not in use. KSB is used to overwrite the coordinate system with the definition and automatically attaches it to the active orientational coordinate system as child

KLN (p. 209) can be used to link KSB(USER) type orientational coordinate systems to other KSB(USER) type coordinate systems or attach them as child to the PI_Base coordinate system.

KSB(USER) type orientational coordinate systems can be activated with KEN (p. 200). DPA SKS (p. 156) reactivates the PI_Base orientational coordinate system that is activated by default; for details see KEN.

KSD (Define Operating Coordinate System By Specifying Values)

Description: Defines a KSD type operating coordinate system.

A KSD type coordinate system can be placed and aligned in space arbitrarily. For more details, see "Work-and-Tool Concept" and in particular "Consideration of KSD and KSF Type Coordinate Systems from the "Work" and "Tool" Perspective" in the "Coordinate Systems for Hexapod Microrobots" document (C887T0007) as well as "Definition of Terms" (p. 3).

The placement of the coordinate system in space is defined by giving offset values for the X, Y, Z axes. The alignment of the coordinate system is defined by giving offset values for the U, V, W axes:

- Rotation around X, i.e., offset specified for U, changes the direction of the Y and Z axes
- Rotation around Y, i.e., offset specified for V, changes the direction of the X and Z axes
- Rotation around Z, i.e., offset specified for W, changes the direction of the X and Y axes

If a KSD type operating coordinate system has been activated with KEN (p. 200):

- The coordinate system determines the position values for the motion platform of the hexapod (get the current position using POS? (p. 238)).
- The coordinates of the center of rotation **cannot** be changed with SPI (p. 245), and the pivot point defined with SPI is **not** used.

If a KSD type coordinate system is not active itself, but is nevertheless linked as a parent of the active operating coordinate system, its offset values are included in the calculation of the offset values for the active operating coordinate system (see KLT? (p. 213)).

The coordinate system is defined in the volatile memory. `WPA SKS` (p. 276) can be used to write the definition to the nonvolatile memory.

Format: KSD <CSName> [{<AxisID> <Offset>}]

Arguments: <CSName> is the name of the coordinate system to be defined.

<AxisID> is one axis of the controller. Possible values: X, Y, Z, U, V, W.

<Offset> is an offset value for placing or aligning the axis; in physical units.

For axes not specified in the KSD command, the offset is set to zero.

Response: None

Notes: When a KSD type operating coordinate system is active:

- In addition to the offset values for the X, Y, Z, U, V, W axes, the coordinate system specifies values for the following settings:
  - NLM (p. 234): Limit for the low end of the axis travel range
  - PLM (p. 236): Limit for the high end of the axis travel range
  - SSL (p. 248): Activation state of the soft limits of the axis
  - If supported by the controller:
    SST: Step size for motion initiated by a manual control unit
- The current settings can be changed with the corresponding commands.
- The current settings can be queried with KLS? (p. 211)
- The current settings are saved for the coordinate system with `WPA SKS` (p. 276).
- For details on triggering motion, see "Commanding Motion" in the document "Coordinate Systems for Hexapod Microrobots" (C887T0007).

Options for defining a coordinate system with KSD:

- <CSName> is a new name. The operating coordinate system is recreated under this name and the ZERO operating coordinate system becomes its parent automatically.
- <CSName> is the name of an existing coordinate system that is not in use. The coordinate system is overwritten by the definition created with KSD.
  - If the overwritten coordinate system was also a KSD type, the chain with its parent and child remains unchanged.
  - If the overwritten coordinate system was not a KSD type, its type is changed to KSD and is attached to ZERO as the new parent.

KLN (p. 209) can be used to link operating coordinate systems of the KSD, KSF, KST, KSW types to other operating coordinate systems of these types or reattach them as successors to the ZERO coordinate system.

`KEN ZERO` and `DPA SKS` re-enable the ZERO operating coordinate system that is enabled by default; for details see KEN (p. 200).

KSF (Define Operating Coordinate System At Current Position)

Description:

Defines a KSF type operating coordinate system at the current position of the hexapod's motion platform.

For more details, see "Work-and-Tool Concept" and in particular "Consideration of KSD and KSF Type Coordinate Systems from the "Work" and "Tool" Perspective" in the "Coordinate Systems for Hexapod Microrobots" document (C887T0007) as well as "Definition of Terms" (p. 3).

If a KSF type operating coordinate system was activated with KEN (p. 200), it determines the position values for the motion platform of the hexapod (get the current position with POS? (p. 238)), and the pivot point defined with SPI (p. 245) is used as center of rotation.

If a KSF type coordinate system is not active itself, but is nevertheless linked as a parent of the active operating coordinate system, its offset values are included in the calculation of the offset values for the active operating coordinate system (see KLT? (p. 213)).

Definition with KSF is only possible when the hexapod is not in motion.

The coordinate system is defined in the volatile memory. `WPA SKS` (p. 276) can be used to write the definition to the nonvolatile memory.

Format:

KSF <CSName>

Arguments:

<CSName> is the name of the coordinate system to be defined.

Response:

None

Notes:

If a KSF type coordinate system is defined, its pivot point coordinates (R, S, T, see SPI (p. 245)) are set to the values valid for the currently active operating coordinate system. If the enabled operating coordinate system does not support the pivot point set using SPI (KSD, KST/KSW types), the pivot point coordinates of the KSF coordinate system are set to R = S = T = 0.

When a KSF type operating coordinate system is active:

- In addition to the offset values for the X, Y, Z, U, V, W axes, the coordinate system specifies values for the following settings:
  - NLM (p. 234): Limit for the low end of the axis travel range
  - PLM (p. 236): Limit for the high end of the axis travel range
  - SSL (p. 248): Activation state of the soft limits of the axis
  - SPI (p. 245): Pivot point coordinates R, S, T

  - If supported by the controller:
    SST: Step size for motion initiated by a manual control unit
- The current settings can be changed with the corresponding commands. The pivot point coordinates can only be changed with SPI if U = V = W = 0 applies to the current position of the platform.
- The current settings can be queried with KLS? (p. 211)
- The current settings are saved for the coordinate system with `WPA SKS` (p. 276).
- For details on triggering motion, see "Commanding Motion" in the document "Coordinate Systems for Hexapod Microrobots" (C887T0007).

Options for defining a coordinate system with KSF:
- <CSName> is a new name. The operating coordinate system is recreated under this name and the ZERO operating coordinate system becomes its parent automatically.
- <CSName> is the name of an existing coordinate system that is not in use. The coordinate system is overwritten by the definition using KSF and automatically attached as successor to the ZERO operating coordinate system.

KLN (p. 209) can be used to link operating coordinate systems of the KSD, KSF, KST, KSW types to other operating coordinate systems of these types or reattach them as successors to the ZERO coordinate system.

`KEN ZERO` and `DPA SKS` re-enable the ZERO operating coordinate system that is enabled by default; for details see KEN (p. 200).

KST (Define "Tool" Operating Coordinate System)

Description: Defines an KST type operating coordinate system ("tool coordinate system") for the work-and-tool concept.

For more details, see "Work-and-Tool Concept" in the "Coordinate Systems for Hexapod Microrobots" document (C887T0007) as well as "Definition of Terms" (p. 3).

The placement of the tool coordinate system is defined by giving offset values for the X, Y, Z axes. The alignment of the tool coordinate system is defined by giving offset values for the U, V, W axes:
- Rotation around X, i.e., offset specified for U, changes the direction of the Y and Z axes
- Rotation around Y, i.e., offset specified for V, changes the direction of the X and Z axes

- Rotation around Z, i.e., offset specified for W, changes the direction of the X and Y axes

When a KST type operating coordinate system is active, the work-and-tool concept is used and the following applies:

- In addition to the tool coordinate system, a work coordinate system must be active. When no KSW type work coordinate system is active, an automatically generated work coordinate system is used as a substitute.
- The coordinates of the center of rotation **cannot** be changed with SPI (p. 245), and the pivot point defined with SPI is **not** used.

If a KST type coordinate system is not active itself, but is nevertheless linked as a parent of the active operating coordinate system, its offset values are included in the calculation of the offset values for the active operating coordinate system (see KLT? (p. 213)).

The coordinate system is defined in the volatile memory. `WPA SKS` (p. 276) can be used to write the definition to the nonvolatile memory.

Format: KST <CSName> [{<AxisID> <Offset>}]

Arguments: <CSName> is the name of the coordinate system to be defined.

<AxisID> is one axis of the controller. Possible values: X, Y, Z, U, V, W.

<Offset> is an offset value for placing or aligning the axis; in physical units.

For axes not specified in the KST command, the offset is set to zero.

Response: None

Notes: When a combination of KSW/KST, or ZERO/KST, or KSW/ZERO type coordinate systems is active:

- In addition to the offset values for the X, Y, Z, U, V, W axes, the combination specifies values for the following settings:
  - NLM (p. 234): Limit for the low end of the axis travel range
  - PLM (p. 236): Limit for the high end of the axis travel range
  - SSL (p. 248): Activation state of the soft limits of the axis
  - If supported by the controller:
    SST: Step size for motion initiated by a manual control unit
- The current settings can be changed with the corresponding commands.
- The current settings can be queried with KLC? (p. 203)
- The current settings are saved with `WPA SKS` (p. 276) for the coordinate system combination.

- For details on triggering motion, see "Commanding Motion" in the document "Coordinate Systems for Hexapod Microrobots" (C887T0007).

Options for defining a coordinate system with KST:

- <CSName> is a new name. The operating coordinate system is recreated under this name and the ZERO operating coordinate system becomes its parent automatically.
- <CSName> is the name of an existing coordinate system that is not in use. The coordinate system is overwritten by the definition created with KST.
  - If the overwritten coordinate system was also a KST type, the chain with its parent and child remains unchanged.
  - If the overwritten coordinate system was not a KST type, its type is changed to KST and it is attached to ZERO as the new parent.

KLN (p. 209) can be used to link operating coordinate systems of the KSD, KSF, KST, KSW types to other operating coordinate systems of these types or reattach them as successors to the ZERO coordinate system.

`KEN ZERO` and `DPA SKS` re-enable the ZERO operating coordinate system that is enabled by default; for details see KEN (p. 200).

KSW (Define "Work" Operating Coordinate System)

Description:

Defines a KSW type operating coordinate system ("work coordinate system") for the work-and-tool concept.

For more details, see "Work-and-Tool Concept" in the "Coordinate Systems for Hexapod Microrobots" document (C887T0007) as well as "Definition of Terms" (p. 3).

The placement of the work coordinate system is defined by giving offset values for the X, Y, Z axes. The alignment of the work coordinate system is defined by giving offset values for the U, V, W axes:

- Rotation around X, i.e., offset specified for U, changes the direction of the Y and Z axes
- Rotation around Y, i.e., offset specified for V, changes the direction of the X and Z axes
- Rotation around Z, i.e., offset specified for W, changes the direction of the X and Y axes

When a KSW type operating coordinate system is active, the work-and-tool concept is used and the following applies:

- In addition to the work coordinate system, a tool coordinate system must be active. When no KST type tool coordinate system is active, an automatically generated tool coordinate system is used as a substitute.
- The coordinates of the center of rotation **cannot** be changed with SPI (p. 245), and the pivot point defined with SPI is **not** used.

If a KSW type coordinate system is not active itself, but is nevertheless linked as a parent of the active operating coordinate system, its offset values are included in the calculation of the offset values for the active operating coordinate system (see KLT? (p. 213)).

The coordinate system is defined in the volatile memory. `WPA SKS` (p. 276) can be used to write the definition to the nonvolatile memory.

Format: KSW <CSName> [{<AxisID> <Offset>}]

Arguments: <CSName> is the name of the coordinate system to be defined.

<AxisID> is one axis of the controller. Possible values: X, Y, Z, U, V, W.

<Offset> is an offset value for placing or aligning the axis; in physical units.

For axes not specified in the KSW command, the offset is set to zero.

Response: None

Notes: When a combination of KSW/KST, or ZERO/KST, or KSW/ZERO type coordinate systems is active:

- In addition to the offset values for the X, Y, Z, U, V, W axes, the combination specifies values for the following settings:
  - NLM (p. 234): Limit for the low end of the axis travel range
  - PLM (p. 236): Limit for the high end of the axis travel range
  - SSL (p. 248): Activation state of the soft limits of the axis
  - If supported by the controller:
    SST: Step size for motion initiated by a manual control unit
- The current settings can be changed with the corresponding commands.
- The current settings can be queried with KLC? (p. 203)
- The current settings are saved with `WPA SKS` (p. 276) for the coordinate system combination.
- For details on triggering motion, see "Commanding Motion" in the document "Coordinate Systems for Hexapod Microrobots" (C887T0007).

Options for defining a coordinate system with KSW:

- <CSName> is a new name. The operating coordinate system is recreated under this name and the ZERO operating coordinate system becomes its parent automatically.

- <CSName> is the name of an existing coordinate system that is not in use. The coordinate system is overwritten by the definition created with KSW.
  - If the overwritten coordinate system was also a KSW type, the chain with its parent and child remains unchanged.
  - If the overwritten coordinate system was not of the KSW type, its type is changed to KSW and it is attached to ZERO as the new parent.

KLN (p. 209) can be used to link operating coordinate systems of the KSD, KSF, KST, KSW types to other operating coordinate systems of these types or reattach them as successors to the ZERO coordinate system.

`KEN ZERO` and `DPA SKS` re-enable the ZERO operating coordinate system that is enabled by default; for details see KEN (p. 200).

LIM? (Indicate Limit Switches)

Description:	Queries whether axes have limit switches.
Format:	LIM? [{<AxisID>}]
Arguments:	<AxisID>: is one axis of the controller
Response:	{<AxisID>"="<uint> LF} where <uint> indicates whether the axis has limit switches (=1) or not (=0).
Troubleshooting:	Illegal axis identifier

MAC (Call Macro Function)

Description:	Calls a macro function. Permits recording, deleting and running macros on the controller.
Format:	MAC <keyword> {<parameter>} in particular: MAC BEG <macroname> MAC DEF <macroname> MAC DEF?

MAC DEL <macroname>
MAC END
MAC ERR?
MAC FREE?
MAC NSTART <macroname> <uint> [{<String>}]
MAC START <macroname> [{<String>}]

Arguments

<keyword> determines which macro function is called. The following keywords and parameters are used:

MAC BEG <macroname>

Starts recording a macro to be named *macroname* on the controller; may not be used in a macro; the commands that follow become the macro. End the recording with MAC END. Note that erroneous macro content cannot be detected by sending the ERR? command.

MAC DEF <macroname>

Sets specified macro as startup macro. This macro will be run automatically after the next switch-on or reboot of the controller. If <macroname> is not specified, the current startup macro selection is canceled.

MAC DEF?

Asks for the startup macro
Response: <macroname>
If a startup macro is not defined, the response is an empty string with the terminating character.

MAC DEL <macroname>

Deletes specified macro.

MAC END

Stops macro recording (cannot become part of a macro)

MAC ERR?

RepoReports the last error that occurred while the macro was running.

Response: <macroname> <uint1>”=”<uint2> <”<”CMD”>”>

where <macroname> is the name of the macro, <uint1> is the line in the macro, <uint2> is the error code and <”<”CMD”>”> is the erroneous command which was sent to the parser.

MAC FREE?

Gets the free memory space for macro recording

Response: <uint> is the number of characters in bytes for which free memory is still available

	MAC NSTART <macroname> <uint> [{<String>}] Repeats the specified macro <uint> times. The macro is re-run each time. <String> stands for the value of a local variable contained in the macro. The sequence of the values in the input must correspond to the numbering of the appropriate local variables, starting with the value of the local variable 1. The individual values must be separated from each other by spaces. A maximum of 256 characters are permitted per command line. <String> can be specified directly or via the value of another variable. refer to "Variables" (p. 131) for further details. MAC START <macroname> [{<String>}] Runs the specified macro once. <String> has the same function as with MAC NSTART.
Response:	None
Troubleshooting:	Macro recording is active (keywords BEG, DEL) or inactive (END) Macro contains a disallowed MAC command
Notes:	During macro recording no macro execution is allowed. When macros are recorded on the ***Controller macros*** tab in PIMikroMove, the `MAC BEG` and `MAC END` commands must be omitted. Macros can contain local and global variables. The names of local variables used in a macro must form a continuous series. Example of permissible designation: 1, 2, 3, 4. Designation with e.g. 1, 2, 5, 6 is not allowed. Further information can be found in the section "Variables" (p. 131). A running macro sends no responses to any interface. Depending on the value of parameter 0x72 (Ignore Macro Error), there are the following possibilities when an error is caused by the running macro: 0 = macro execution is aborted 1 = the error is ignored and macro execution will be continued Irrespective of the parameter setting, MAC ERR? always reports the last error that occurred during a macro execution. The following commands provided by the C-887 can only be used in macros: ADD (p. 150), CPY (p. 152), DEL (p. 154), JRC (p. 199), MEX (p. 227) and WAC (p. 264).

A macro can start another macro. The maximum number of nesting levels is 10. A macro can call itself to form an infinite loop.

Any commands can be sent from the command line when a macro is running. The macro content and motion commands received from the command line can overwrite each other, and only the last motion command will be executed, irrespective of its source.

Macro execution can be stopped with #24 (p. 146) and STP (p. 252).

It is not possible to run several macros simultaneously. Only one macro can be executed at a time.

A macro cannot be deleted while it is running.

You can query with #8 (p. 145) if a macro is currently running on the controller.

Warning: The number of write cycles of nonvolatile memory is limited.

MAC? (List Macros)

Description:	Lists macros or content of a specified macro.
Format:	MAC? [<macroname>]
Arguments	<macroname>: name of the macro where the content is to be listed; if not specified, the names of all stored macros are listed.
Response:	<string> If <macroname> was specified, <string> is the content of this macro; If <macroname> was not specified, <string> is a list with the names of all stored macros
Troubleshooting:	Macro <macroname> not found

MAN? (Get Help String For Command)

Description:	Shows a detailed help text for individual commands.
Format:	MAN? <CMD>

Arguments: <CMD> is the command mnemonic of the command for which the help text is to be displayed (see below).

Response: A string that describes the command.

Notes: A detailed help text can be displayed for the following GCS commands: FDR, SIC, WPA, WAV, WTR, WGO, WAV?

Example: Send: `MAN? WPA`

Receive:

```
WPA <Password> [{<ElementID> <PamID>}] Save
Parameters To Non-Volatile Memory
#AvailablePasswords
<Pswd> <Param_Setting>
100 All Parameters, Settings of Coordinate
System Configuration and Assignment of Axes A, B
101 All Parameters
SKS Settings of Coordinate System Configuration
A12 Assignment of Axes A, B
end of help
```

MEX (Stop Macro Execution Due To Condition)

Description: Stops running the macro due to a specified condition of the following type: a specified value is compared with a queried value according to a specified rule.

Can only be used in macros.

When the macro interpreter accesses this command, the condition is checked. If it is true, the current macro is stopped; otherwise the macro continues to run with the next line. Should the condition be fulfilled later, the interpreter will ignore it.

See also the WAC command (p. 264).

Format: MEX <CMD?> <OP> <Value>

Arguments	<CMD?> is one query command in its usual syntax. The response has to be a single value and not more. For an example see below. <OP> is the operator to be used. The following operators are possible: = <= < > >= != Important: There must be a blank space before and after the operator! <Value> is the value that is compared with the response to <CMD?>.
Response:	None

MOV (Set Target Position)

Description:	Sets an absolute target position for the specified axis.
Format:	MOV {<AxisID> <Position>}
Arguments:	<AxisID> is one axis of the controller. <Position> is the absolute target position in physical units.
Response:	none
Troubleshooting:	▪ Target position outside of the current workspace. ▪ Illegal axis identifier See section "Commandable Elements" (p. 25) for more information. ▪ Servo mode is OFF for one of the axes specified. ▪ The reference move has not been successfully completed for at least one axis.
Notes:	The servo mode must be switched on when this command is used (closed-loop operation). The physical unit for specifying the <Position> can be queried with PUN? (p. 238) In order to determine whether a motion has been completed, it is recommended to send #5 (p. 143). The motion can be aborted by #24 (p. 146), STP (p. 252) and HLT (p. 193).

The following is valid for the axes of the hexapod (X, Y, Z, U, V, W):

- Before the start of any motion, it is checked whether the motion platform can actually reach the commanded target position. You can query whether the target position can be reached with VMO? (p. 263)
- Depending on the setting of the ***Trajectory Source*** parameter (ID 0x19001900), the dynamics profile for the axes of the hexapod (X, Y, Z, U, V, W) is defined by one of the following two sources:
  - Profile generator of the C-887
  - Cyclic transfer of target positions by consecutive MOV commands

For more information, see "Hexapod Motion" (p. 31).

Example 1:

Send: MOV X 10 U 5

Note: Axis X moves to 10 (target position in mm), axis U moves to 5 (target position in °)

Example 2:

Send: MOV X 4 Y 2.3 Z -3 U -5.3 V 3 W 1

Note: Target positions can be given for all six axes of the hexapod with a single motion command.

Example 3:

Send: MOV Z 100

Send: ERR?

Receive: 7

Note: The axis does not move. The error code "7" in the response to the ERR? command (p. 164) indicates that the target position given in the motion command is out of limits

MOV? (Get Target Position)

Description: Returns last valid commanded target position.

Format: MOV? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller

Response: {<AxisID>"="<float> LF}

where

<float> is the last commanded target position in physical units

Troubleshooting: Illegal axis identifier

Notes:	The target position can be changed by different sources; see "Motion of the hexapod" (p. 31). MOV? gets the commanded positions. Use POS? (p. 238) to get the current positions.

MRT (Set Target Relative In Tool Coordinate System)

Description:	Moves the specified axis relative in the tool coordinate system. For more details, see "Work-and-Tool Concept" in the "Coordinate Systems for Hexapod Microrobots" document (C887T0007) as well as "Definition of Terms" (p. 3). While the target position to be approached is being determined from the values for <Distance>, the translation is calculated first and then the rotation. Servo mode must be switched on for the commanded axis prior to using this command (closed-loop operation).
Format:	MRI {<AxisID> <Distance>}
Arguments:	<AxisID> is one axis of the controller. Possible values: X, Y, Z, U, V, W. <Distance> gives the distance that the axis is to move; the sum of the distance and the last commanded target position is set as the new target position (in physical units).
Response:	None
Troubleshooting:	▪ Target position outside of the current workspace. ▪ The ***Trajectory Source*** parameter (ID 0x19001900) is set to 1 (but must be set to 0 when MRT is used). ▪ Servo mode is OFF for one of the axes specified. ▪ The reference move has not been successfully completed for at least one axis.
Notes:	When the work-and-tool concept is not applied, motion is performed with MRT in the tool coordinate system, which exists according to the active operating coordinate system (see "Consideration of KSD and KSF Type Coordinate Systems" from the "Work" and "Tool" perspective in the "Coordinate Systems for Hexapod Microrobots" document (C887T0007)). The physical unit for specifying the <Distance> can be queried with PUN? (p. 238)

In order to determine whether motion has been completed, it is recommended to send #5 (p. 143).

Motion can be aborted by #24 (p. 146), STP (p. 252), and HLT (p. 193).

MRW (Set Target Relative In Work Coordinate System)

Description: Moves the specified axis relative in the work coordinate system.

For more details, see "Work-and-Tool Concept" in the "Coordinate Systems for Hexapod Microrobots" document (C887T0007) as well as "Definition of Terms" (p. 3).

While the target position to be approached is being determined from the values for <Distance>, the translation is calculated first and then the rotation.

Servo mode must be switched on for the commanded axis prior to using this command (closed-loop operation).

Format: MRW {<AxisID> <Distance>}

Arguments: <AxisID> is one axis of the controller. Possible values: X, Y, Z, U, V, W.

<Distance> gives the distance that the axis is to move; the sum of the distance and the last commanded target position is set as the new target position (in physical units).

Response: None

Troubleshooting:

- Target position outside of the current workspace.
- The ***Trajectory Source*** parameter (ID 0x19001900) is set to 1 (but must be set to 0 when MRW is used).
- Servo mode is OFF for one of the axes specified.
- The reference move has not been successfully completed for at least one axis.

Notes: When the work-and-tool concept is not applied, motion is performed with MRW in the work coordinate system, which exists according to the active operating coordinate system (see "Consideration of KSD and KSF Type Coordinate Systems" from the "Work" and "Tool" perspective in the "Coordinate Systems for Hexapod Microrobots" document (C887T0007)).

The physical unit for specifying the <Distance> can be queried with PUN? (p. 238)

In order to determine whether motion has been completed, it is recommended to send #5 (p. 143).

Motion can be aborted by #24 (p. 146), STP (p. 252), and HLT (p. 193).

MVR (Set Target Relative To Current Position)

Description:	Moves the specified axis relative to the last commanded target position.
Format:	MVR {<AxisID> <Distance>}
Arguments:	<AxisID> is one axis of the controller. <Distance> specifies the distance that the axis is to move; the sum of the distance and the last commanded target position is set as the new target position (in physical units).
Response:	none
Troubleshooting:	▪ Target position outside of the current workspace. ▪ Illegal axis identifier See section "Commandable Elements" (p. 25) for more information. ▪ Servo mode is OFF for one of the axes specified. ▪ The reference move has not been successfully completed for at least one axis.
Notes:	The servo mode must be switched on when this command is used (closed-loop operation). The physical unit for specifying the <Distance> can be queried with PUN? (p. 238) In order to determine whether a motion has been completed, it is recommended to send #5 (p. 143). The motion can be aborted by #24 (p. 146), STP (p. 252) and HLT (p. 193). The following is valid for the axes of the hexapod (X, Y, Z, U, V, W): ▪ Before the start of any motion, it is checked whether the motion platform can actually reach the commanded target position. You can query whether the target position can be reached with VMO? (p. 263) ▪ When the ***Trajectory Source*** parameter (ID 0x19001900) is set to 1, the dynamics profile must be defined by consecutive MOV commands. MVR is not permitted. For more information, see "Hexapod Motion" (p. 31).

Example:	Send: MOV X 0.5 Note: This is an absolute motion. Send: POS? X Receive: X=0.500000 Send: MOV? X Receive: X=0.500000 Send: MVR X 2 Note: This is a relative motion. Send: POS? X Receive: X=2.500000 Send: MVR X 2000 Note: New target position of axis X would exceed the motion range. Command is ignored, i.e. the target position remains unchanged, and the axis does not move. Send: MOV? X Receive: X=2.500000 Send: POS? X Receive: X=2.500000

NAV (Set Number of Readout Values to be Averaged)

Description:	Determines the number of readout values of the analog input that are averaged.
Format:	NAV {<AnalogInputID> <NumberOfReadings>}
Arguments:	<AnalogInputID> is the identifier of the analog input channel. <NumberOfReadings> is the number of readout values of the analog signal
Troubleshooting:	Number of readout values too high Illegal identifier of the analog input channel.
Notes:	The default value for <NumberOfReadings> is 1; the permissible range of values is 1 to 10000. NAV can reduce the influence of noise at the analog input on queries (TAV? (p. 254), TAD? (p. 254)) or on scanning procedures (e.g., AAP (p. 147)). The greater the noise of the analog input signal, the higher the number of readout values of the analog signal should be set with NAV for averaging. For fast alignment routines, i.e., all routines that are defined with the FDR and FDG commands and started with the FRS command, the readout values of the analog input are **not** averaged. NAV has no

effect on these routines.

Example: Send: NAV 2 200

Send: TAV? 2

Note: The response to the TAV? command 2 is the average of 200 values read from analog input channel 2.

NAV? (Get Number of Readings to be Averaged)

Description: Queries the number of readout values of the analog input used for averaging.

Format: NAV? [{<AnalogInputID>}]

Arguments: <AnalogInputID> is the identifier of the analog input channel

Response: {<AnalogInputID>"="<int> LF}

where

<int> is the number of readout values.

The response consists of a line feed when the controller does not contain an analog input channel.

NLM (Set Low Position Soft Limit)

Description: Limits the low end of the axis travel range in closed-loop operation ("soft limit").

Format: NLM {<AxisID> <LowLimit>}

Arguments: <AxisID> is one axis of the controller

<LowLimit> is the limit position for the low end of the travel range, in physical units

Response: None

Notes:	For the axes of the motion platform of the hexapod, the default values for NLM are specified by the enabled operating coordinate system (with the work-and-tool concept, from the combination of two operating coordinate systems). The value set with NLM must be smaller than the current position value. Only negative values are allowed for axes X, Y, Z, U, V and W. The soft limits are activated and deactivated with SSL (p. 248). Soft limits can only be set when the axis is not moving (query with #5 (p. 143)). The physical unit in which <LowLimit> is to be given can be queried with PUN? (p. 238). When the pivot point is changed with SPI (p. 247), the soft limits for the rotational axes U, V and W are not adjusted.
Example:	Send: NLM? X Receive: X = -22.5 Send: POS? X Receive: X = -10 Send: NLM X -5 Send: ERR? Receive: 27 - (error 27 - "Soft limit out of range") Send: NLM? X Receive: X = -22.5

NLM? (Get Low Position Soft Limit)

Description:	Get the position "soft limit" which determines the low end of the axis travel range in closed-loop operation.
Format:	NLM? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller
Response:	{<AxisID>"="<LowLimit> LF} where <LowLimit> is the limit position for the low end of the travel range, in physical units.

ONT? (Get On-Target State)

Description:	Queries the on-target state of the specified axis. If all arguments are left out, queries state of all axes.
Format:	ONT? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller.
Response:	{<AxisID>"="<uint> LF} where <uint> = "1" when the specified axis has reached the target value, otherwise "0".
Troubleshooting:	Illegal axis identifier
Notes:	ONT? gets the status of the axes of the hexapod motion platform (X to W) as well as of axes A and B. When querying with ONT?, the C-887 only checks whether the end of the dynamics profile for the axes of the hexapod's motion platform (X to W) has been reached but **not** the actual motion status. The C-887 shows the actual motion status of all axes when queried #5 (p. 143); furthermore, the status register bits indicate the status of the hexapod struts and axes A and B, for details, see "Motion Status, Settling Window, Settling Time" (p. 42).

PLM (Set High Position Soft Limit)

Description:	Limits the high end of the axis travel range in closed-loop operation ("soft limit").
Format:	PLM {<AxisID> <HighLimit>}
Arguments:	<AxisID> is one axis of the controller <HighLimit> is the limit position for the high end of the travel range, in physical units.
Response:	None

Notes:	For the axes of the motion platform of the hexapod, the default values for PLM are specified by the enabled operating coordinate system (with the work-and-tool concept, from the combination of two operating coordinate systems). The value set with PLM must be greater than the current position value. Only positive values are allowed for axes X, Y, Z, U, V and W. The soft limits are activated and deactivated with SSL (p. 248). Soft limits can only be set when the axis is not moving (query with #5 (p. 143)). The physical unit in which <HighLimit> is to be given can be queried with PUN? (p. 238). When the pivot point is changed with SPI (p. 247), the soft limits for the rotational axes U, V, and W are not adjusted.
Example:	Send: PLM? X Receive: X = 22.5 Send: POS? X Receive: X = 10 Send: PLM X 5 Send: ERR? Receive: 27 - (error 27 - "Soft limit out of range") Send: PLM? X Receive: X = 22.5

PLM? (Get High Position Soft Limit)

Description:	Gets the position "soft limit" which determines the high end of the axis travel range in closed-loop operation.
Format:	PLM? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller

Response:	{<AxisID>"="<HighLimit> LF} where <HighLimit> is the limit position for the high end of the travel range, in physical units.

POS? (Get Real Position)

Description:	Queries the current axis position. If no arguments are specified, the current position of all axes is queried.
Format:	POS? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller.
Response:	{<AxisID>"="<float> LF} where <float> is the current axis position in physical units.
Troubleshooting:	Illegal axis identifier
Notes:	This command is identical in function to #3 (p. 142) that should be preferred when the controller is doing time-consuming tasks. The current position of axes X, Y, Z, U, V and W is calculated from the measured positions of the individual struts. Between the switching on of the controller and the referencing of the hexapod with FRF (p. 174), the current position of the hexapod and axes A and B is unknown. Nevertheless, the response to POS? gives the position value 0 for all axes. The physical unit for specifying the axis position can be queried with PUN? (p. 238)

PUN? (Get Axis Unit)

Description:	Queries the current unit of the axis. If all arguments are omitted, the current unit for all axes is queried.

Format: PUN? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller.

Response: {<AxisID>"="<string> LF}

where

<string> is the current unit of the axis.

Troubleshooting: Illegal axis identifier

Note: The following units are used for the axis positions:
X, Y, Z: Millimeters
U, V, W: Degrees
A, B: Millimeters or degrees
The units cannot be changed.

RBT (Reboot System)

Description: Reboots system. The controller behaves the same as after switching on.

Format: RBT

Arguments: none

Response: none

RMC? (List Running Macros)

Description: Lists macros which are currently running.

Format: RMC?

Arguments: None

Response: {<macroname> LF}

where

<macroname> is the name of one macro which is saved on the controller and currently running. The response is an empty line when no macro is running.

RON? (Get Reference Mode)

Description:	Queries referencing method of specified axes.
Format:	RON? [{ <AxisID>}]
Arguments:	<AxisID> is one axis of the controller.
Response:	{<AxisID>"="<ReferenceOn> LF} where <ReferenceOn> is the currently selected referencing method for the axis
Troubleshooting:	Illegal axis identifier
Notes:	RON? always returns 1. For axes whose position is measured by incremental sensors, this means: ▪ Motion commands are only executed after successful referencing. ▪ Referencing must be done by a reference move. With the C-887, the reference move is made to the reference switch (start with FRF (p. 174)).

RTR (Set Record Table Rate)

Description:	Sets the record table rate, i.e., the number of cycles to be used in data recording operations. Settings larger than 1 make it possible to cover longer time periods.
Format:	RTR <RecordTableRate>
Arguments:	<RecordTableRate> is the record table rate to be used for recording operations (unit: number of cycles), must be an integer value larger than zero.
Response:	None
Notes:	RTR sets the value of the ***Data Recorder Table Rate*** parameter (ID 0x16000000) in the volatile memory. The setting can be saved in the nonvolatile memory of the C-887 with the WPA command (p. 276), so that it is available immediately after the C-887 is switched on or rebooted. Values between 1 and 10000 are possible. By default, the value of the parameter is set to 10 (corresponds to 1 kHz) . If the C-887 does fast alignment routines (p. 3), it sets the record rate table to the factor 4. The duration of the recording can be calculated as follows:

Recording duration = cycle time * RTR value *number of points

where

the cycle time of the data recorder is 100 µs

the number of points for the C-887 is a maximum of 262144

For further information, see "Data Recorder" (p. 97).

RTR? (Get Record Table Rate)

Description:	Queries the current record table rate, i.e., the number of cycles used in data recording operations.
Format:	RTR?
Arguments:	None
Response:	<RecordTableRate> is the table rate used for recording operations (unit: number of cycles).
Note:	Gets the ***Data Recorder Table Rate*** parameter value in the volatile memory (ID 0x16000000). For further information, see "Data Recorder" (p. 97).

SAI? (Get List Of Current Axis Identifiers)

Description:	Queries the axis identifiers. Refer also to "Commandable Elements" (p. 25).
Format:	SAI? [ALL]
Arguments:	[ALL] is optional. For controllers that allow deactivating the axis, [ALL] ensures that the response also includes the axes that are "deactivated".
Response:	{<AxisID> LF} <AxisID> is one axis of the controller.
Note:	Axes A and B can be "deactivated" by assigning the "NOSTAGE" positioner type with the CST command (p. 153). Deactivated axes are not shown. Exception: The responses to the CST? (p. 154) and SAI? ALL commands include the deactivated axes.

SCT (Set Cycle Time)

Description: Determines the cycle time for executing a dynamics profile.

Used in the cyclic transfer of target positions, see also "Motion of the hexapod" (p. 31).

Format: SCT "T" <CycleTime>

Arguments: "T" is the required keyword for the argument <CycleTime>

<CycleTime> is the cycle time in ms, format: float.

Troubleshooting: Permissible value range exceeded

Notes: The permissible range of values for <CycleTime> is 1 to 10000 ms; the default value is 1 ms.

The cycle time set with SCT is only effective when the ***Trajectory Source*** parameter (ID 0x19001900) has the value 1 (= the dynamics profile is defined by consecutive MOV commands).

The cycle time is used to calculate the velocity during motion so that the specified points of the dynamics profile are reached exactly at the end of the time interval.

When the ***Trajectory Execution*** parameter (ID 0x19001901) has the value 1, intermediate storage of the dynamics profile points guarantees that the dynamics profile is executed at the specified time intervals.

When the ***Trajectory Execution*** parameter (ID 0x19001901) has the value 0, the MOV commands are executed immediately after being sent.

➢ Make sure that the MOV commands are sent in the time intervals specified in the cycle time so that the dynamics profile can be maintained.

Example: Send: SCT T 30

Sets the cycle time to 30 ms for executing a dynamics profile that is defined by consecutive MOV commands.

SCT? (Get Cycle Time)

Description: Gets the current cycle time for running a defined dynamics profile.

Format: SCT? [<T>]

Arguments: "T" serves as a keyword and can be omitted for the query.

Response: T=<float> LF

<float> is the cycle time in ms.

SGA (Set Gain)

Description: Determines the gain value for the given analog input channel.

Format: SGA {<AnalogInputID> <Gain>}

Arguments: <AnalogInputID> is the identifier of the analog input channel.

<Gain> is the gain factor

Troubleshooting: Illegal value

Note: The default value for <Gain> is 1; other values are not permitted. The SGA command is only present for compatibility reasons.

For the identifiers of the analog input channels, see "Commandable Elements" (p. 25).

SGA? (Get Gain)

Description: Gets the gain value for the specified analog input channel.

Format: SGA? [{<AnalogInputID>}]

Arguments: <AnalogInputID> is the identifier of the analog input channel, see SGA (p. 243)

Response: {<AnalogInputID>"="<int> LF}

where

<int>is the gain value of the analog input channel.

SPA (Set Volatile Memory Parameters)

Description:	Sets a parameter of the specified element in the volatile memory (RAM) to a specific value. Parameter changes are lost when the controller is switched off or rebooted.
Format:	SPA {<ItemID> <PamID> <PamValue>}
Arguments:	<ItemID> is the element for which a parameter is changed in volatile memory. See below for details. <PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details. <PamValue> is the value to which the specified parameter of the specified element is set.
Response:	None **Note that this command is for setting hardware-specific parameters. Wrong values may lead to improper operation or damage of your hardware!** With HPA? (p. 193) you can obtain a list of the available parameters.
Troubleshooting:	Illegal element identifier, wrong parameter ID, value out of range, command level too low for write access Parameter values can only be set when the axis is not in motion (query with #5 (p. 143)).
Element IDs and parameter IDs available:	The element can be an axis, a hexapod strut (drive or sensor), an input signal channel, or the whole system; the element type depends on the parameter. See "Parameter Overview" (p. 302) for the element type concerned; for element identifiers, see "Commandable Elements" (p. 25). Valid parameter IDs are given in "Parameter Overview" (p. 302).
Example:	Send: SPA 1 0x16000000 8 Note: Sets the record table rate for the controller to 8, parameter ID written in hexadecimal format Send: SPA 1 369098752 2 Note: Sets the record table rate for the controller to 2, parameter ID written in decimal format

SPA? (Get Volatile Memory Parameters)

Description:	Queries the value of a parameter of a specified element from volatile memory (RAM). You can obtain a list of the available parameters with HPA? (p. 193).
Format:	SPA? [{<ItemID> <PamID>}]
Arguments:	<ItemID> is the element for querying a parameter in volatile memory. See below for details. <PamID> is the parameter identifier, can be written in hexadecimal or decimal format. See below for details.
Response:	{<ItemID> <PamID>"="<PamValue> LF} where <PamValue> is the value of the specified parameter for the specified element
Troubleshooting:	Illegal element identifier, wrong parameter ID
Element IDs and parameter IDs available:	The element can be an axis, a hexapod strut (drive or sensor), an input signal channel, or the whole system; the element type depends on the parameter. See "Parameter Overview" (p. 302) for the element type concerned; for element identifiers, see "Commandable Elements" (p. 25). Valid parameter IDs are given in "Parameter Overview" (p. 302).

SPI (Set Pivot Point)

Description:	Moves the pivot point out of the origin of the coordinate system in the X and/or Y direction and/or Z direction when the following conditions are met: ▪ The ZERO coordinate system or a KSF type coordinate system is activated as the operating coordinate system. ▪ For the rotation coordinates of the motion platform, U = V = W = 0 applies SPI sets the pivot point coordinates in the volatile memory.
Format:	SPI {<PPCoordinate> <Position>}

Arguments:	<PPCoordinate> is a pivot point coordinate, see below. <Position> is the value of the pivot point coordinate, see below.
Response:	None
Troubleshooting:	At least one of the rotation coordinates U, V and W is not equal to 0 The active coordinate system is not ZERO and not of the KSF type.
Notes:	<PPCoordinate> can be R, S and T. X, Y and Z can also be used as alias identifiers for R, S and T. <Position> is given in mm. The default values are specified by the enabled operating coordinate system (only for coordinate systems of the KSF and ZERO types). For further information on the pivot point, see "Definition of Terms" (p. 3). If the pivot point is moved with SPI, the possible travel ranges for the rotational axes U, V and W change and thus the workspace as well. The following values are **not** adapted to the changed travel ranges, however: ▪ Responses to TMN? (p. 255) and TMX? (p. 256) ▪ Soft limits that are set with NLM (p. 234) and PLM (p. 236) Use VMO? (p. 263) to query whether a target position can be reached.
Example:	Send: SPI? Receive: R=0 S=0 T=0 Send: SPI S 2 Send: SPI? Receive: R=0 S=2 T=0 Send: SPI Z 2 Send: SPI? Receive: R=0 S=2 T=2

SPI? (Get Pivot Point)

Description:	Gets the pivot point coordinates.
Format:	SPI? [{<PPCoordinate>}]
Arguments:	<PPCoordinate> is a pivot point coordinate, see below
Response:	{<PPCoordinate>"="<Position> LF} where <Position> is the value of the pivot point coordinate in physical units.
Note:	<PPCoordinate> can be R, S and T. X, Y and Z can also be used as alias identifiers for R, S and T.

SRG? (Query Status Register Value)

Description:	Returns register values for queried elements and registers.
Format:	SRG? [{<ItemID> <RegisterID>}]
Arguments:	<ItemID> is the element for querying a register. See below for details. <RegisterID> is the ID of the specified register; see below for available registers.
Response:	{<ItemID><RegisterID>"="<Value> LF} where <Value> is the value of the register; see below for more details.
Possible IDs and response values:	<ItemID> can be 1 to 8. The IDs 1 to 8 correspond in this order to the hexapod struts 1 to 6 and to axes A and B. <RegisterID> is always 1. <Value> is the bit-mapped response. See "Status Registers for Hexapod Struts and Axes A and B" (p. 358) for the description of bits.

Notes: The status register bits queried with the SRG? command can also be recorded with the C-887's data recorder (p. 97), record option 80 (status register of axis).

You can query the system status register (p. 359) with the #4 (p. 143) and STA? (p. 251) commands.

SSL (Set Soft Limit)

Description: Activates or deactivates the soft limits that are set with NLM (p. 234) and PLM (p. 236).

Format: SSL {<AxisID> <SoftLimitsOn>}

Arguments: <AxisID> is one axis of the controller

<SoftLimitsOn> is the status of the soft limits:

0 = soft limits deactivated
1 = soft limits activated

Response: None

Note: For the axes of the motion platform of the hexapod, the default values for SSL are specified by the enabled operating coordinate concept (with the work-and-tool system, from the combination of two operating coordinate systems).

Soft limits can only be activated/deactivated when the axis is not moving (query with #5 (p. 143)).

Example: Send: SSL X 1
The soft limits for axis X are activated.

Send: SSL Y 0 Z 1 W 1
The soft limits are deactivated for the axis Y and activated for axes Z and W.

SSL? (Get Soft Limit Status)

Description:	Gets the status of the soft limits that are set with NLM (p. 234) and PLM (p. 236). If all arguments are omitted, the status is queried for all axes.
Format:	SSL? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller
Response:	{<AxisID>"="<SoftLimitsOn> LF} where <SoftLimitsOn> is the status of the soft limits: 0 = soft limits deactivated 1 = soft limits activated
Troubleshooting:	Illegal axis identifier

SSN? (Get Device Serial Number)

Description:	Gets the serial number of the C-887.
Format:	SSN?
Arguments:	None
Response:	<SerialNumber> is the serial number of the device.

SST (Set Step Size)

Description:	Sets the distance ("step size") for motions of the given axis that are triggered by a manual control unit.
Format:	SST {<AxisID> <StepSize>}
Arguments:	<AxisID> is one axis of the controller <StepSize> is the distance, format: float
Response:	None
Troubleshooting:	Illegal value Illegal axis identifier

Note:	The physical unit for specifying the <StepSize> can be queried with PUN? (p. 238) The permissible range of values for <StepSize> is 0 to 0.5; the default value is 0.01. The step size set with SST is used for moving the axes of the hexapod's motion platform (X, Y, Z, U, V, W) triggered manually by a C-887.MC2 or C-887.MC control unit (p. 22).
Example:	Send: SST Y 0.002 U 0.05 Send: SST? Receive: X=0.01 Y=0.002 Z=0.01 U=0.05 V=0.01 W=0.01 Send: SST X 0.09 W 0.09 Send: SST? Receive: X=0.09 Y=0.002 Z=0.01 U=0.05 V=0.01 W=0.09

SST? (Get Step Size)

Description:	Gets the distance ("step size") for motions of the given axis that are triggered by a manual control unit.
Format:	SST? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller
Response:	{<AxisID>"="<StepSize> LF} where <StepSize> is the distance in physical units, see SST (p. 249).

STA? (Query Status Register Value)

Description:	Requests system status information.
Format:	STA?
Arguments:	None
Response:	The response is bit-mapped. See below for the individual codes.
Notes:	The response is the sum of the codes in hexadecimal format. For the description of the bits and an example, see "System Status Register" (p. 359). The function of this command is identical to #4 (p. 143). Additional status information that **cannot** be queried with #4 and STA?: The C-887 has a status register (p. 358) for each of the struts 1 to 6 of the hexapod and for axes A and B. You can query the bits of these registers with the SRG? command (p. 247) and record with the C-887's data recorder (p. 97), record option 80 (status register of axis).

STE (Start Step And Response Measurement)

Description:	Starts a step and records the step response for the specified axis. The data recorder configuration, i.e., the assignment of data sources and record options to the recorder tables, can be set with DRC (p. 157). The recorded data can be read with the DRR? command (p. 160).
Format:	STE <AxisID> <Amplitude>
Arguments:	<AxisID> is one axis of the controller <Amplitude> is the size of the step. See below for details.
Response:	None
Troubleshooting:	The target position resulting from the specified step height is out of limits.
Notes:	A "step" consists of a relative move of the specified amplitude. The step is executed relative to the current position. The physical unit for specifying the <Amplitude> can be queried with PUN? (p. 238) The following is valid for the axes of the hexapod (X, Y, Z, U, V, W):

- Before the start of any motion, it is checked whether the motion platform can actually reach the commanded target position. You can query whether the target position can be reached with VMO? (p. 263)
- When the ***Trajectory Source*** parameter (ID 0x19001900) is set to 1, the dynamics profile must be defined by consecutive MOV commands. STE is not permitted.

For more information, see "Hexapod Motion" (p. 31).

STP (Stop All Axes)

Description: Stops all axes abruptly. See the notes below for further details.

Sets error code to 10.

This command is identical in function to #24 (p. 146).

Format: STP

Arguments: None

Response: None

Troubleshooting: Communication breakdown

Notes: STP stops all axis motion caused by motion commands, fast alignment routines, scanning procedures, or wave generator output, and stops the reference move.

STP stops macros.

After the axes are stopped, their target positions are set to their current positions.

SVO (Set Servo Mode)

Description: Sets the servo mode for specified axes (open-loop or closed-loop operation).

Format: SVO {<AxisID> <ServoState>}

Arguments: <AxisID> is one axis of the controller

<ServoState> can have the following values:
0 = servo mode off (open-loop operation)
1 = servo mode on (closed-loop operation)

Response: None

Troubleshooting: Illegal axis identifier

Notes: Servo mode is always switched on or off together for the axes of the motion platform of the hexapod (X, Y, Z, U, V, W). For this reason, it is sufficient to enter a single axis, e.g.,
SVO X 1, to set the servo mode for the axes of the motion platform

Servo mode is switched on automatically in the following cases:

- Switching on or rebooting the C-887 – servo mode is switched on for all axes
- For the axes of the hexapod's motion platform (X, Y, Z, U, V, W): A reference move is started with FRF (p. 174)
- For axes A and B: Assigning a positioner type with CST (p. 153)

Axis motion can only be triggered when servo mode is switched on.

Servo mode can only be switched off with SVO when the axis is not moving (query with #5 (p. 143)).

SVO? (Get Servo Mode)

Description: Queries the servo mode for the axes specified.

If arguments are not specified, queries the servo mode of all axes.

Format: SVO? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller.

Response: {<AxisID>"="<ServoState> LF}

where

<ServoState> is the current servo mode for the axis:
0 = servo mode off (open-loop operation)
1 = servo mode on (closed-loop operation)

Troubleshooting: Illegal axis identifier

TAC? (Tell Analog Channels)

Description:	Gets the number of installed analog lines.
Format:	TAC?
Arguments:	None
Response:	<uint> indicates the total number of analog lines (inputs and outputs).
Note:	All analog lines included in the response to TAC? are analog input channels. For the identifiers of the analog input channels, see "Commandable Elements" (p. 25).

TAD? (Get ADC Value Of Input Signal)

Description:	Get the current value from the specified input signal channel's A/D converter. Using this command it is possible to check for sensor overflow.
Format:	TAD? [{<InputSignalID>}]
Arguments:	<InputSignalID> is one input signal channel of the controller
Response:	{<InputSignalID>"="<uint> LF} where <uint> is the current A/D value, dimensionless
Notes:	The TAD? response represents the digitized signal value without filtering and linearization. The greater the noise of the analog input signal, the higher the number of readout values of the analog signal should be set with NAV (p. 233) for averaging. For the identifiers of the analog input channels, see "Commandable Elements" (p. 25).

TAV? (Get Analog Input Voltage)

Description:	Get voltage at analog input.
Format:	TAV? [{<AnalogInputID>}]

Arguments: <AnalogInputID> is the identifier of the analog input channel; see below for details.

Response: {<AnalogInputID>"="<float> LF}

where

<float> is the current voltage at the analog input in volts

Note: The greater the noise of the analog input signal, the higher the number of readout values of the analog signal should be set with NAV (p. 233) for averaging.

For the identifiers of the analog input channels, see "Commandable Elements" (p. 25).

TMN? (Get Minimum Commandable Position)

Description: Get the minimum commandable position in physical units.

Format: TMN? [{ <AxisID>}]

Arguments: <AxisID> is one axis of the controller

Response {<AxisID>"="<float> LF}

where

<float> is the minimum commandable position in physical units

Notes: The travel ranges in X, Y, Z, U, V, W are dependent on each other. Depending on the current position of the motion platform of the hexapod, the actually available travel range for axes X, Y, Z, U, V, and W can be smaller than the travel range displayed in the response to TMN? The response to TMN? only corresponds to the actually available travel range of an axis when the following conditions are met:

- All other axes are at zero position.
- The factory-set coordinate systems are active.
- The defaults for the pivot point coordinates apply.

If the pivot point is moved with SPI (p. 245), the available travel ranges for the rotational axes U, V and W change. The values for the minimum commandable positions are **not** adapted to the changed travel ranges, however.

Use VMO? (p. 263) to query whether a target position can be reached.

If you want to work with user-defined coordinate systems, move the pivot point, and/or execute combined motions of several axes: Use TRA? (p. 257) to get the maximum absolute position that can be commanded when the platform of the hexapod moves along a specified direction vector.

The physical unit in which the minimum commandable position is given can be queried with PUN? (p. 238)

TMX? (Get Maximum Commandable Position)

Description: Get the maximum commandable position in physical units.

Format: TMX? [{ <AxisID>}]

Arguments: <AxisID> is one axis of the controller

Response {<AxisID>"="<float> LF}

where

<float> is the maximum commandable position in physical units

Notes: The travel ranges in X, Y, Z, U, V, W are dependent on each other. Depending on the current position of the motion platform of the hexapod, the actually available travel range for axes X, Y, Z, U, V, and W can be smaller than the travel range displayed in the response to TMX? The response to TMX? only corresponds to the actually available travel range of an axis when the following two conditions are met:

- All other axes are at zero position.
- The factory-set coordinate systems are active.
- The defaults for the pivot point coordinates apply.

If the pivot point is moved with SPI (p. 245), the available travel ranges for the rotational axes U, V and W change. The values for the maximum commandable positions are **not** adapted to the changed travel ranges, however.

Use VMO? (p. 263) to query whether a target position can be reached.

If you want to work with user-defined coordinate systems, move the pivot point, and/or execute combined motions of several axes: Use TRA? (p. 257) to get the maximum absolute position that can be commanded when the platform of the hexapod moves along a specified direction vector.

The physical unit in which the maximum commandable position is given can be queried with PUN? (p. 238)

TNR? (Get Number of Record Tables)

Description:	Queries the number of data recorder tables currently available on the controller.
Format:	TNR?
Arguments:	none
Response	<uint> is the number of data recorder tables which are currently available
Notes:	The response indicates the value of the ***Data Recorder Channel Number*** parameter (ID 0x16000300). For more information see "Data Recording" (p. 97).

TRA? (Get Maximum Commandable Position For Direction Vector)

Description:	Queries the maximum absolute position that can be commanded when the platform of the hexapod moves along the direction vector that is defined by the specified axis components. The maximum commandable position is calculated from the current position and it can be queried only if the platform of the hexapod is not in motion. Note: "Maximum" refers to the amount of the position value. Therefore, in this description the largest possible displacement in a negative direction is referred to as the "maximum" position as well (and not as the "minimum" position).
Format:	TRA? {<AxisID> <Component>}
Arguments:	<AxisID> is one axis of the controller. Possible values: X, Y, Z, U, V, W. <Component> is the component of the axis on the direction vector. Must be different from zero for at least one queried axis. Can have a negative sign.

	Axes not specified in the query have no component on the direction vector and are not included in the response.
Response:	{<AxisID>"="<float> LF} where <float> is the maximum commandable absolute position for the axis when the platform of the hexapod moves along the specified direction vector; in physical units.
Notes:	Included in the calculation are the current settings for the soft limits (see NLM (p. 234), PLM (p. 236), SSL (p. 248)) and for the pivot point defined with SPI (see SPI (p. 245)) if it is used by the active operating coordinate system. Due to rounding, the response to TRA? may indicate a position that cannot be reached. Commanding such a position fails and generates error code 7 ("Position out of limits"). Therefore, you can limit the response to TRA? with a factor so that only positions that can actually be commanded are displayed: ➢ Use SPA to set the ***Reduction Factor for TRA? Response*** parameter (0x19006000) to a suitable value between 0 and 1. The physical unit where the maximum commandable position is specified, can be queried with PUN? (p. 238).
Example:	The motion platform is to move in the direction of the vector (X, Z) = (2, 4). The PLM command sets the upper soft limit for the X axis to the value 1. Send: TRA? X 2 Z 4 The response indicates the maximum absolute position that can be approached in the direction of the vector (X, Z) = (2, 4) from the current position: X=1.00000 Z=1.99869

TRS? (Indicate Reference Switch)

Description:	Indicates whether axes have a reference switch with direction sensing.
Format:	TRS? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller

Response:	{<AxisID>"="<uint> LF} where <uint> indicates whether the axis has a direction-sensing reference switch (=1) or not (=0).
Troubleshooting:	Illegal axis identifier

TWG? (Get Number of Wave Generators)

Description:	Queries the number of wave generators available in the controller.
Format:	TWG?
Arguments:	None
Response	<uint> is the number of wave generators which are available

VAR (Set Variable Value)

Description:	Sets a variable to a certain value. Local variables can be set using VAR in macros only. See “Variables” (p. 131) for more details on local and global variables. The variable is present in RAM only.
Format:	VAR <Variable> <String>
Arguments:	<Variable> is the name of the variable whose value is to be set. <String> is the value to which the variable is to be set. If not specified, the variable is deleted. The value can be specified directly or via the value of a variable. Refer to “Variables” (p. 131) for more details on conventions regarding variable names and values.
Response:	None
Example:	It is possible to set the value of one variable (e.g. TARGET) to that of another variable (e.g. SOURCE):

```
VAR TARGET ${SOURCE}
```

Use curly brackets if the name of the variable is longer than one character:

```
VAR A ONE
VAR VARB TWO
VAR $A 1
VAR ${VARB} 2
VAR $VARB 2 // this will result in an unwanted behavior
VAR?
A=ONE
VARB=TWO
ONE=1
TWO=2 // ${VARB}: is replaced by its value "TWO"
ARB=2 // $VARB: $V is replaced by its (empty) value
```

See ADD (p. 150) for another example.

VAR? (Get Variable Values)

Description: Gets values of variables.

If VAR? is combined with CPY (p. 152), JRC (p. 199), MEX (p. 227) or WAC (p. 264), the response to VAR? has to be a single value and not more.

Refer to “Variables” (p. 131) for more details on local and global variables.

Format: VAR? [{<Variable>}]

Arguments: <Variable> is the name of the variable to be queried. Refer to "Variables" (p. 131) for more details on name conventions.

All global variables present in RAM are listed if <Variable> is not specified.

Response: {<Variable>”=”<String>LF}

where

<String> gives the value to which the variable is set.

Note: Within a macro, VAR? can only be meaningfully used in combination with CPY (p. 152), JRC (p. 199), MEX (p. 227) or WAC (p. 264).

VEL (Set Closed-Loop Velocity)

Description: Set velocity of specified axes.

Format: VEL {<AxisID> <Velocity>}

Arguments: <AxisID> is one axis of the controller.

<Velocity> is the velocity value in physical units/s.

Response: None

Troubleshooting: Illegal axis identifiers

Notes: The command is only permissible for axes A and B.

<Velocity> must be ≥ 0. The upper limit depends on the positioner type that is assigned to the axes A and B (see CST (p. 153)).

The velocity set with VEL is saved in volatile memory (RAM) only.

VEL? (Get Closed-Loop Velocity)

Description: Queries the commanded velocity.

If no arguments are specified, queries the value of all axes.

Format: VEL? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller.

Response: {<AxisID>"="<float> LF}

where

<float> is the currently valid velocity value commanded in physical units per second.

VER? (Get Versions Of Firmware And Drivers)

Description: Gets the versions of the firmware of the C-887 as well as of further components like, for example, drivers and libraries.

Format: VER?

Arguments: None

Response: {<string1>":" <string2> [<string3>]LF}

where

<string1> is the name of the component;
<string2> is the version information of the component <string1>;
<string3> is an optional note.

Example: Meaning of the response lines for the C-887:

`FW: V 2.3.1.1` Version of the firmware component FW

`Macro: V 0.15.0.0` Version of the macro functionality

`OS: V #1 SMP PREEMPT RT Fri Jan 22 12:54:58 CET 2016` Version of the operating system of the C-887

`Hexdata: V 1.0.0.0` Version of the configuration file with the geometrical data for the hexapod to which the C-887 is adapted

`PIStages: V 200.7 19/10/2017 15:1:28` Version of the PIStages2.dat positioner database saved to the C-887

`FPGA/DSP:  V 0.28 V 0.97` Version of the motor control

`ADC:  C887E0015 SN116054450 V3 11/10/2016` Version of the AD converter for the analog inputs 5 and 6; models C-887.521, .523, .531, .533 only

`C-887.MC: V 256 SN 118015195` Version of the manual control unit; only when a control unit is connected to the C-887

`IDChip: H-811.I2 SN117007310 21/2/2017` Information on the type, serial number, and manufacturing date of the hexapod saved on the ID chip of the hexapod

`Collision: V 2.15` Version of the library for the optionally available PIVeriMove hexapod software for collision checking; is also displayed when the option is not activated

VLS (Set System Velocity)

Description: Sets the velocity for the motion platform of the hexapod.

Format: VLS <SystemVelocity>

Arguments: <SystemVelocity> is the velocity value in physical units.

Response: None

Notes: The unit for <SystemVelocity> is mm/s.

The lower limit for <SystemVelocity> is a result of minimum incremental motion of the hexapod (depending on model) and is specified by the value of the ***Minimum System Velocity*** parameter (ID 0x19001501). The parameter can be queried with SPA? (p. 245)

The velocity can only be set with VLS when the hexapod is not moving (axes X, Y, Z, U, V, W; get with #5 (p. 143)).

The velocity can be set for axes A and B with VEL (p. 261).

VLS? (Get System Velocity)

Description: Gets the velocity of the motion platform of the hexapod that is set with VLS (p. 262).

Format: VLS?

Arguments: None

Response: <SystemVelocity> is the velocity value in physical units, see VLS.

VMO? (Virtual Move)

Description: Checks whether the motion platform of the hexapod can approach a specified position from the current position.

VMO? does not trigger any motion.

Format: VMO? {<AxisID> <Position>}

Arguments: <AxisID> is an axis of the controller, see below

<Position> is a target position value to be checked

Response: <uint> indicates whether the motion platform can approach the position resulting from the given target position values:
0 = specified position cannot be approached
1 = specified position can be approached

Notes: Axes X, Y, Z, U, V, W are permissible.

VMO? checks the following:

- Are the nodes of the calculated profile and the target position outside of the travel range limits that can be queried with TMN? (p. 255) and TMX? (p. 256) or TRA? (p. 257)?
- Have the soft limits set with NLM (p. 234) and PLM (p. 236) been activated with SSL (p. 248), and if yes, are the nodes and the target position outside of these soft limits?
- Are the individual struts able to move the platform to the required nodes and the specified target position?

- When a configuration for collision avoidance has been stored in the C-887 with the optionally available PIVeriMove hexapod software for collision checking: Do collisions occur between the following groups?
  - Surroundings including base plate of the hexapod
  - Hexapod struts
  - Motion platform of the hexapod incl. load

In order to receive a reliable response, do not send VMO? until after a successful reference move (start with FRF (p. 174)) and only when the hexapod is not moving (query with #5 (p. 143)).

VST? (Get Connectable Stages)

Description:	Lists positioner types suitable for connecting to the C-887.
Format:	VST?
Arguments:	None
Response:	<string> is a list of all positioner types included in the available positioner databases and suitable for use with the C-887.
Notes:	You can find further information on positioner databases in "Operating Parameters for Axes A and B" (p. 30). The positioner types listed by VST? can be assigned to axes A and B with the CST command (p. 153).

WAC (Wait For Condition)

Description:	Waits until a specified condition of the following type occurs: a specified value is compared with a queried value according a specified rule. Can only be used in macros. See also the MEX command (p. 227).
Format:	WAC <CMD?> <OP> <value>

Arguments <CMD?> is one query command in its usual notation. The response has to be a single value and not more. For an example see below.

<OP> is the operator to be used. The following operators are possible:
= <= < > >= !=
Important: There must be a blank space before and after the operator!

<value> is the value to be compared with the response to <CMD?>.

Response: None

Example: Send:

```
MAC BEG LPMOTION
MVR 1 1
 WAC ONT? 1 = 1
MVR 1 -1
 WAC ONT? 1 = 1
MAC START LPMOTION
MAC END
MAC START LPMOTION
```

Note: Macro LPMOTION is first recorded and then started. WAC ONT? 1 = 1 waits until the response to ONT? 1 is 1=1. To form an infinite loop, the macro calls itself.

WAV (Set Waveform Definition)

Description: Defines a waveform of given type for given wave table.

To allow flexible definition, a waveform (wave table contents) can be built up by stringing "segments" together. Each segment is defined with a separate WAV command. A segment can be added to the existing wave table contents with the <AppendWave> argument (see below). To change individual segments or to modify their order, the entire waveform must be re-created segment-by-segment.

A segment can be based on predefined waveforms (see the <WaveType> argument below).

Waveforms cannot be changed while they are being output by a wave generator. Before a waveform is modified with WAV, the wave generator output from the associated wave table must be stopped first.

The waveform values are absolute values.

The duration of an output cycle for the waveform can be calculated as follows:

Output duration = servo cycle time * WTR value * number of points

where

the servo cycle time for the C-887 is given by the parameter 0x0E000200 (in seconds)

the WTR value (output rate of the wave generator) indicates the number of servo cycles for which the output of a waveform point lasts; integer multiple of 10 (minimum and default: 10, maximum 1000)

the number of points corresponds to the wave table length (sum of the lengths of all segments in this table)

For more information, see "Wave Generator" (p. 99).

Format: WAV <WaveTableID> <AppendWave> <WaveType> <WaveTypeParameters>

Arguments: <WaveTableID> is the wave table identifier.

<AppendWave> can be "X" or "&":
"X" clears the wave table and starts writing at the first point in the table.
"&" attaches the defined segment to the existing wave table contents in order to extend the waveform.

<WaveType> The type of curve used to define the segment. This can be one of
"PNT" (user-defined curve)
"SIN_P"(inverted cosine curve)
"RAMP" (ramp curve)
"LIN" (single scan line curve)

<WaveTypeParameters> stands for the parameters of the curve:

For "PNT":

<WaveStartPoint> <WaveLength> {<WavePoint>}

<WaveStartPoint>: The index of the starting point. Must be 1.

<WaveLength>: The number of points to be written in the wave table (= segment length).

<WavePoint>: The value of one single point.

For "SIN_P":

<SegLength> <Amp> <Offset> <WaveLength> <StartPoint> <CurveCenterPoint>

<SegLength>: The length of the wave table segment in points. Only the number of points specified by <SegLength> will be written to the wave table. If the <SegLength> value is larger than the <WaveLength> value, the missing points in the segment are filled with the endpoint value of the curve.

<Amp>: The amplitude of the sine curve.

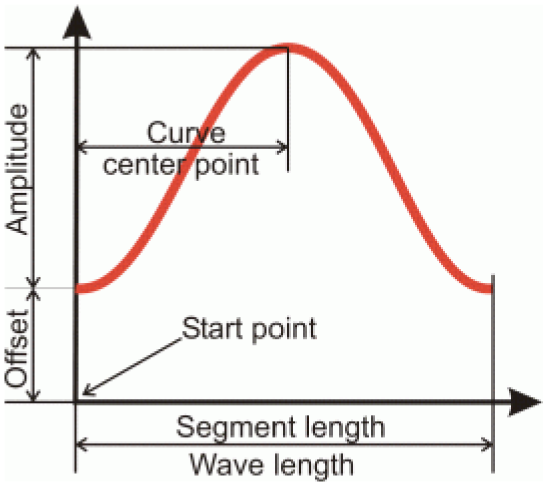
<Offset>: The offset of the sine curve.

<WaveLength>: The length of the sine curve in points.

<StartPoint>: The index of the starting point of the sine curve in the segment. Specifies the phase shift. Lowest possible value is 0.

<CurveCenterPoint>: The index of the center point of the sine curve. Determines if the curve is symmetrical or not. Lowest possible value is 0.

Example (refer to "Defining the Waveform" (p. 102) for further examples):



For "RAMP":

<SegLength> <Amp> <Offset> <WaveLength> <StartPoint> <SpeedUpDown> <CurveCenterPoint>

<SegLength>: The length of the wave table segment in points. Only the number of points specified by <SegLength> will be written to the wave table. If the <SegLength> value is larger than the <WaveLength> value, the missing points in the segment are filled with the endpoint value of the curve.

<Amp>: The amplitude of the ramp curve.

<Offset>: The offset of the ramp curve.

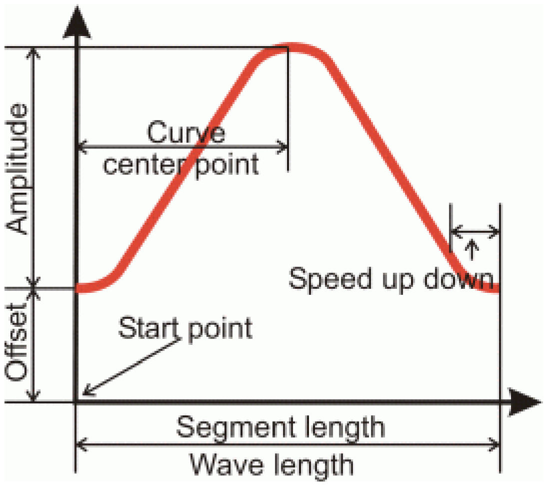
<WaveLength>: The length of the ramp curve in points.

<StartPoint>: The index of the starting point of the ramp curve in the segment. Specifies the phase shift. Lowest possible value is 0.

<SpeedUpDown>: The number of points for acceleration and delay.

<CurveCenterPoint>: The index of the center point of the ramp curve. Determines if the curve is symmetrical or not. Lowest possible value is 0.

Example (refer to "Defining the Waveform" (p. 102) for further examples):



For "LIN":

<SegLength> <Amp> <Offset> <WaveLength> <StartPoint> <SpeedUpDown>

<SegLength>: The length of the wave table segment in points. Only the number of points specified by <SegLength> will be written to the wave table. If the <SegLength> value is larger than the <WaveLength> value, the missing points in the segment are filled with the endpoint value of the curve.

<Amp>: The amplitude of the scan line.

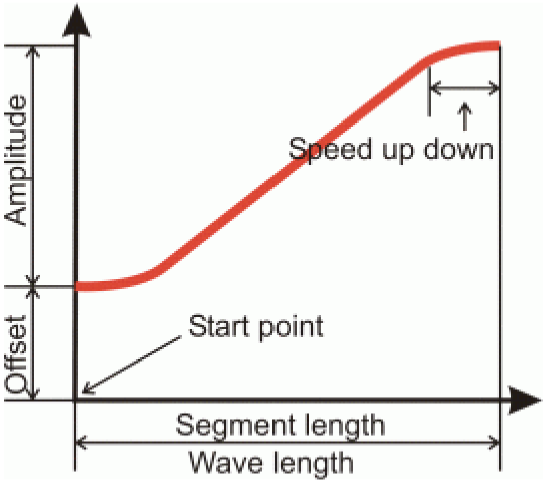
<Offset>: The offset of the scan line.

<WaveLength>: The length of the single scan line curve in points.

<StartPoint>: The index of the starting point of the scan line in the segment. Lowest possible value is 0.

<SpeedUpDown>: The number of points for acceleration and delay.

Example (refer to "Defining the Waveform" (p. 102) for further examples):



Response: None

Troubleshooting: Invalid wave table identifier

The total number of points for the waveform (which may consist of several segments) exceeds the available number of memory points.

Notes:

When a waveform is defined with WAV, the C-887 does **not** check the target positions and the resulting velocities. The check is only done during the wave generator output (p. 113). If motion is not possible, the C-887 stops the wave generator output abruptly and therefore the motion.

To define suitable waveforms with WAV and prevent the wave generator output from aborting, observe the following:

- Calculate the waveform externally as precisely as possible (e.g., with NI LabVIEW, MATLAB or Python) before you transfer the waveform definition to the C-887 with WAV.
- Note that the waveform influences the velocity during the motions. Define the waveform so that the specifications of the connected mechanics are adhered to during wave generator output. The velocity is limited by the following factors, among others:
  - Mechanics type
  - Combination of the axes to be moved
  - Current settings for coordinate system and center of rotation
  - Amplitude of the motion
- If the waveform is to be output periodically (i.e., more than one output cycle in succession), the position and velocity at the end of the waveform must be identical to those at the beginning of the waveform.
- If you create a waveform from several segments, the initial position and initial velocity of a segment to be attached must be adapted to the end position and end velocity of the preceding segment.
- Define the waveform so that it can be output with the default output rate. For higher output rates, the C-887 interpolates missing position values, which means that the output waveform may no longer correspond precisely to the definition.

WAV? (Get Waveform Definition)

Description: Queries the value of a wave parameter for a specified wave table.

Refer to "Wave Generator" (p. 99) for more information.

Format: WAV? [{<WaveTableID> <WaveParameterID>}]

Arguments: <WaveTableID> is the wave table identifier.

<WaveParameterID> is the wave parameter ID:
1 = Current wave table length as a number of points

Response: {<WaveTableID> <WaveParameterID>"="<float> LF}

where

<float> depends on the <WaveParameterID>; specifies the current number of waveform points in the wave table for <WaveParameterID> = 1

Troubleshooting: Invalid wave table identifier

WCL (Clear Wave Table Data)

Description: Clears the content of the specified wave table.

As long as a wave generator is running, it is not possible to clear the connected wave table.

Refer to "Wave Generator" (p. 99) for more information.

Format: WCL {<WaveTableID>}

Arguments: <WaveTableID> is the wave table identifier.

Response: None

WGC (Set Number Of Wave Generator Cycles)

Description: Sets the number of output cycles for the specified wave generator (the output itself is started with WGO (p. 272)).

Refer to "Wave Generator" (p. 99) for more information.

Format: WGC {<WaveGenID> <Cycles>}

Arguments: <WaveGenID> is the wave generator identifier

<Cycles> is the number of wave generator output cycles.

Response: None

Notes: With <WaveGenID> = 0, all wave generators are addressed.

If cycles = 0, then the waveform is output without limitation until it is stopped by WGO or #24 (p. 146) or STP (p. 252) or HLT (p. 193).

WGC? (Get Number Of Wave Generator Cycles)

Description: Queries the number of output cycles set for the specified wave generator.

Refer to "Wave Generator" (p. 99) for more information.

Format: WGC? [{<WaveGenID>}]

Arguments: <WaveGenID> is the wave generator identifier

Response: {<WaveGenID>"="<Cycles> LF}

where

<Cycles> is the number of wave generator output cycles set with WGC (p. 271).

WGO (Set Wave Generator Start/Stop Mode)

Description: Starts and stops the specified wave generator. When the wave generator output is started, a data recording cycle automatically starts.

All wave generators whose output has to be active at the same time must be started in the same command.

For the axes of the motion platform of the hexapod whose wave generator is **not** started, the last valid target position is always commanded.

The number of output cycles can be limited by WGC (p. 271).

With the WTR command (p. 278), you can lengthen the individual output cycles of the waveform.

The wave generator output is also continued when the PC software with which it was started is exited.

The #9 command can be used to query the current activation state of the wave generators. WGO? queries the last start options commanded for the wave generator. Further status information on the wave generator can be queried with WGS? (p. 274).

For more information, see "Wave Generator" (p. 99).

Format: WGO {<WaveGenID> <StartMode>}

Arguments: <WaveGenID> is the wave generator identifier

<StartMode> is the start mode for the specified wave generator and can be given in hexadecimal or decimal format. Possible values:

0: wave generator output is stopped. You can also stop the wave generator output with #24 (p. 146) or STP (p. 252) or HLT (p. 193).

bit 0 = 0x1 (hex format) or 1 (decimal format):
start wave generator output immediately, synchronized by servo cycle

Response: None

Troubleshooting: Invalid wave generator identifier

There is no wave table connected to the wave generator. Connect a wave table with WSL (p. 278).

When the wave generator output is active, commands for starting or configuring motion and executing corresponding macros are **not** permitted.

During the wave generator output, the C-887 continuously checks whether the motion is still possible. In the following cases, the C-887 stops the motion abruptly and sets an error code:

- The target positions to be output cannot be achieved.
- The necessary velocity cannot be achieved.
- The motion would cause a collision.

➢ Use the `WGS?` command to get the current status of the wave generators, especially the index of the waveform points at which an error has occurred.

➢ Get the error code of the last error that occurred with the `ERR?` command (p. 164).

WGO? (Get Wave Generator Start/Stop Mode)

Description: Queries the start/stop mode of the specified wave generator.

Refer to "Wave Generator" (p. 99) for more information.

Format: WGO? [{<WaveGenID>}]

Arguments: <WaveGenID> is the wave generator identifier

Response:	{<WaveGenID>"="<StartMode> LF} where <StartMode> is the last commanded start mode of the wave generator, in decimal format. Refer to WGO (p. 272) for further information.
Notes:	<StartMode> is not only 0 when `WGO <WaveGenID> 0` is sent but also when the wave generator output has ended or #24 (p. 146) or STP or HLT (p. 193) has been used for stopping.

WGR (Starts Recording In Sync With Wave Generator)

Description:	Starts the data recording when the wave generator is active. Refer to "Wave Generator" (p. 99) and "Data Recorder" (p. 97) for more information.
Format:	WGR
Arguments:	None
Response:	None
Notes:	After WGR has been sent, the data recording starts with the next output cycle of the wave generator. The recorded data can be read with DRR? (p. 160). Starting the wave generator output with WGO (p. 272) starts an initial data recording cycle at the same time. For further trigger options for starting the data recording, see DRT (p. 163).

WGS? (Get Status Information On Wave Generator)

Description:	Gets status information on the given wave generator. For more information, see "Wave Generator" (p. 99).
Format:	WGS? [<WaveGenID> [<ItemID>]]

Arguments:	<WaveGenID> is the wave generator identifier <ItemID> is the identifier of a variable to be queried. Possible identifiers: STATUS Gets the status of the wave generator. Possible response values, in hexadecimal format: 0x0 = wave generator not active (output not running) 0x1 = Wave generator active (output running) ITERATIONS Gets the number of output cycles since the last start of the wave generator. Stopping the wave generator stops the counter. The counter is reset when the wave generator is started again with the WGO command. ERRORTYPE Gets the error code of the last error that occurred during the output. Possible response values: 0 = No error occurred 7 = Position out of limits 8 = Velocity out of limits 91 = Move not possible, would cause collision ERRORINDEX Gets the index of the waveform point at which the error occurred.
Response:	{<WaveGenID> <ItemID>"="<Value> LF} where <Value> is the current value of the queried variable.

WMS? (Get Maximum Number of Wave Table Points)

Description:	Gets the number of available memory points for the given wave table.
Format:	WMS? [{<WaveTableID>}]
Arguments:	<WaveTableID> is the wave table identifier.

Response {<WaveTableID>"="<NumberOfPoints> LF}

where

<NumberOfPoints> is the number of memory points that are available for the wave table (sum of the points already defined with WAV (p. 265) and the still unused points).

WPA (Save Parameters To Nonvolatile Memory)

Description: Writes the current settings from the volatile to the nonvolatile memory.

The settings saved with WPA are automatically loaded to the volatile memory when the C-887 is switched on or rebooted.

Note: Incorrect settings can cause the system to malfunction. Make sure that the current settings are correct before you execute the WPA command.

Settings in the volatile memory not saved using WPA will be lost when the C-887 is switched off or rebooted, or when settings are restored.

Format: WPA <Pswd> [{<ElementID> <PamID>}]

Arguments: <Pswd> is the password for writing to the nonvolatile memory. See below for details.

<ElementID> is the element for which a parameter is to be saved from the volatile to the nonvolatile memory. See below for details.

<PamID> is the parameter identifier, can be written in hexadecimal or decimal format. See below for details.

Response: None

Troubleshooting: Illegal element identifier, wrong parameter ID, invalid password

Notes: When parameter settings are to be saved:

- Parameter values can be changed in the volatile memory with the SPA command (p. 244).
- An element can be an axis, a hexapod strut, an input signal channel, or the entire system. The element type depends on the parameter. For further information, see "Adapting Settings" (p. 299).
- You can get a list of all available parameters with HPA? (p. 193) Valid parameter IDs are also specified in "Parameter Overview" (p. 302).

In addition to parameter settings, WPA can write the settings for coordinate systems to the nonvolatile memory (for details, see the table below).

Saving with WPA does **not** overwrite the default settings, which can be restored with DPA (p. 156).

Note: Avoid switching the C-887 off during the WPA procedure.

Valid passwords for writing to the nonvolatile memory:

100	Saves the currently valid values of all parameters, the currently valid settings for coordinate systems (for details, see password SKS), and the current assignment of positioner types to axes A and B
101	Saves the currently valid parameter values. Specification of <ItemID> and <PamID> is optional.
SKS	When the SKS password is used, <ItemID> and <PamID> are not needed. Saves the currently valid settings for coordinate systems: ▪ Properties of the coordinate systems and combinations of coordinate systems in the volatile memory; see KLS? and KLC? ▪ Activation state of coordinate systems, see KEN ▪ Linking of coordinate systems, see KLN When ZERO is active: The current values for NLM, PLM, SSL, and SPI are not saved. This ensures that KEN ZERO reactivates the default settings fully for the operating coordinate system.
A12	Assignment of positioner types to axes A and B:

Valid passwords can be queried with `MAN? WPA`.

WSL (Set Connection Of Wave Table To Wave Generator)

Description: Wave table selection: connects a wave table to a wave generator or disconnects the selected generator from any wave table.

Two or more generators can be connected to the same wave table, but a generator cannot be connected to more than one wave table.

Deleting wave table content with WCL (p. 271) has no effect on the WSL settings.

As long as a wave generator is running, it is not possible to change its wave table connection.

Refer to "Wave Generator" (p. 99) for more information.

Format: WSL {<WaveGenID> <WaveTableID>}

Arguments: <WaveGenID> is the wave generator identifier

<WaveTableID> is the wave table identifier. If <WaveTableID> = 0, the selected generator is disconnected from any wave table.

Response: None

WSL? (Get Connection Of Wave Table To Wave Generator)

Description: Queries current wave table connection settings for the specified wave generator.

Refer to "Wave Generator" (p. 99) for more information.

Format: WSL? [{<WaveGenID>}]

Arguments: <WaveGenID> is the wave generator identifier

Response: {<WaveGenID>"="<WaveTableID> LF}

where

<WaveTableID> is the wave table identifier. If <WaveTableID> = 0, no wave table is connected to the wave generator.

WTR (Set Wave Generator Table Rate)

Description: Sets wave generator table rate and interpolation type.

Format:	WTR {<WaveGenID> <WaveTableRate> <InterpolationType>}
Arguments:	<WaveGenID> is the wave generator identifier. See below for details. <WaveTableRate> is the wave generator table rate (unit: number of servo cycles); must be an integer value that is greater than zero <InterpolationType> Available interpolation types: See below.
Response:	None
Notes:	Different output rates can be set for the individual wave generators of the C-887. The output rate is set to the same value for all wave generators when <WaveGenID> has the value zero. The individual output cycles of the waveform can be lengthened with the WTR command. The duration of an output cycle for the waveform can be calculated as follows: Output duration = servo cycle time * WTR value * number of points where the servo cycle time for the C-887 is given by the parameter 0x0E000200 (in seconds) the WTR value (output rate of the wave generator) indicates the number of servo cycles for which the output of a waveform point lasts; integer multiple of 10 (minimum and default: 10, maximum 1000) Number of points is the length of the waveform (i.e., the length of the wave table) WTR also sets the interpolation type used for the wave generator output with an output rate > 10. Note: Linear interpolation can prevent position jumps, but the waveform can then no longer be differentiated at the nodes. Greater jumps of velocity and acceleration can result and induce oscillation in the mechanics. For this reason, use the default output rate (10) if possible. For more information, see "Wave Generator" (p. 99). An application example can be found under "Configuring the Wave Generator" (p. 111).

Interpolation types available:

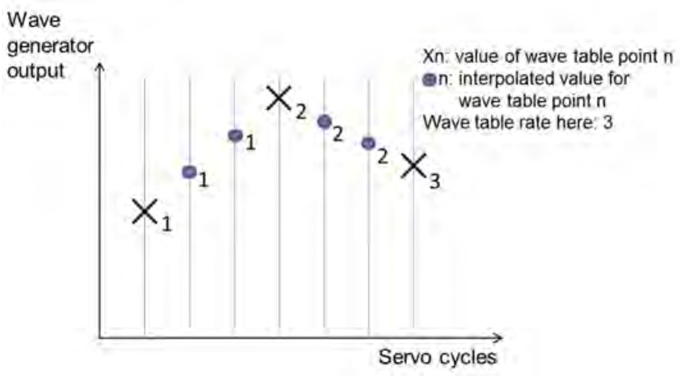
The following interpolation types are available:
0 = No interpolation
1 = Straight line (default)

Examples:

Note: The C-887 does **not** support the output rate given in the following examples (3). The examples are only to show the principle of interpolation.

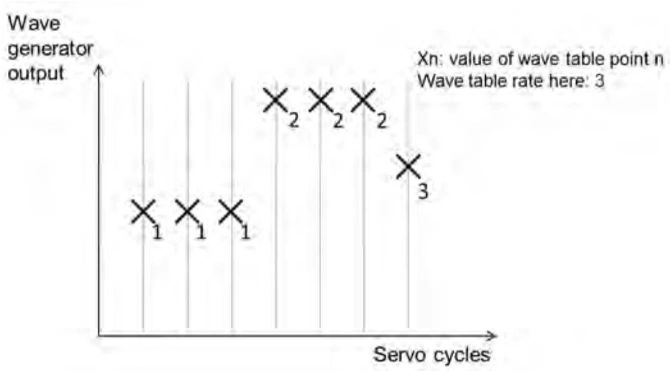
Interpolation type: Straight line (1; default)
The output rate for wave generator 5 is set to 3, with linear interpolation:

WTR 5 3 1



Interpolation type: No interpolation (0)
The output rate for wave generator 5 is set to 3, without interpolation:

WTR 5 3 0



WTR? (Get Wave Generator Table Rate)

Description: Queries the current wave generator table rate and the used interpolation type.

Refer to "Wave Generator" (p. 99) for more information. An application example can be found under "Configuring a Wave Generator" (p. 111).

Format: WTR? [{<WaveGenID>}]

Arguments: <WaveGenID> is the wave generator identifier

Response: {<WaveGenID>"="<WaveTableRate> <InterpolationType> LF}

where

<WaveTableRate> is the wave generator table rate (unit: Number of servo cycles)

<InterpolationType> is the interpolation type applied to outputs between wave table points when the output rate is higher than the minimum value. Refer to WTR (p. 278) for available interpolation types.

8.5 Error Codes

The error codes listed here are those of the PI General Command Set. As such, some may be not relevant to your controller and will simply never occur.

Controller Errors

0	PI_CNTR_NO_ERROR	No error
1	PI_CNTR_PARAM_SYNTAX	Parameter syntax error
2	PI_CNTR_UNKNOWN_COMMAND	Unknown command
3	PI_CNTR_COMMAND_TOO_LONG	Command length out of limits or command buffer overrun
4	PI_CNTR_SCAN_ERROR	Error while scanning
5	PI_CNTR_MOVE_WITHOUT_REF_OR_NO_SERVO	Unallowable move attempted on unreferenced axis, or move attempted with servo off
6	PI_CNTR_INVALID_SGA_PARAM	Parameter for SGA not valid
7	PI_CNTR_POS_OUT_OF_LIMITS	Position out of limits
8	PI_CNTR_VEL_OUT_OF_LIMITS	Velocity out of limits
9	PI_CNTR_SET_PIVOT_NOT_POSSIBLE	Attempt to set pivot point while U,V and W not all 0

10	PI_CNTR_STOP	Controller was stopped by command
11	PI_CNTR_SST_OR_SCAN_RANGE	Parameter for SST or for one of the embedded scan algorithms out of range
12	PI_CNTR_INVALID_SCAN_AXES	Invalid axis combination for fast scan
13	PI_CNTR_INVALID_NAV_PARAM	Parameter for NAV out of range
14	PI_CNTR_INVALID_ANALOG_INPUT	Invalid analog channel
15	PI_CNTR_INVALID_AXIS_IDENTIFIER	Invalid axis identifier
16	PI_CNTR_INVALID_STAGE_NAME	Unknown stage name
17	PI_CNTR_PARAM_OUT_OF_RANGE	Parameter out of range
18	PI_CNTR_INVALID_MACRO_NAME	Invalid macro name
19	PI_CNTR_MACRO_RECORD	Error while recording macro
20	PI_CNTR_MACRO_NOT_FOUND	Macro not found
21	PI_CNTR_AXIS_HAS_NO_BRAKE	Axis has no brake
22	PI_CNTR_DOUBLE_AXIS	Axis identifier specified more than once
23	PI_CNTR_ILLEGAL_AXIS	Illegal axis
24	PI_CNTR_PARAM_NR	Incorrect number of parameters
25	PI_CNTR_INVALID_REAL_NR	Invalid floating point number
26	PI_CNTR_MISSING_PARAM	Parameter missing
27	PI_CNTR_SOFT_LIMIT_OUT_OF_RANGE	Soft limit out of range
28	PI_CNTR_NO_MANUAL_PAD	No manual pad found
29	PI_CNTR_NO_JUMP	No more step-response values
30	PI_CNTR_INVALID_JUMP	No step-response values recorded
31	PI_CNTR_AXIS_HAS_NO_REFERENCE	Axis has no reference sensor
32	PI_CNTR_STAGE_HAS_NO_LIM_SWITCH	Axis has no limit switch
33	PI_CNTR_NO_RELAY_CARD	No relay card installed
34	PI_CNTR_CMD_NOT_ALLOWED_FOR_STAGE	Command not allowed for selected stage(s)
35	PI_CNTR_NO_DIGITAL_INPUT	No digital input installed
36	PI_CNTR_NO_DIGITAL_OUTPUT	No digital output configured
37	PI_CNTR_NO_MCM	No more MCM responses
38	PI_CNTR_INVALID_MCM	No MCM values recorded
39	PI_CNTR_INVALID_CNTR_NUMBER	Controller number invalid
40	PI_CNTR_NO_JOYSTICK_CONNECTED	No joystick configured
41	PI_CNTR_INVALID_EGE_AXIS	Invalid axis for electronic gearing, axis can not be slave
42	PI_CNTR_SLAVE_POSITION_OUT_OF_RANGE	Position of slave axis is out of range
43	PI_CNTR_COMMAND_EGE_SLAVE	Slave axis cannot be commanded directly when electronic gearing is enabled
44	PI_CNTR_JOYSTICK_CALIBRATION_FAILED	Calibration of joystick failed

45	PI_CNTR_REFERENCING_FAILED	Referencing failed
46	PI_CNTR_OPM_MISSING	OPM (Optical Power Meter) missing
47	PI_CNTR_OPM_NOT_INITIALIZED	OPM (Optical Power Meter) not initialized or cannot be initialized
48	PI_CNTR_OPM_COM_ERROR	OPM (Optical Power Meter) Communication Error
49	PI_CNTR_MOVE_TO_LIMIT_SWITCH_FAILED	Move to limit switch failed
50	PI_CNTR_REF_WITH_REF_DISABLED	Attempt to reference axis with referencing disabled
51	PI_CNTR_AXIS_UNDER_JOYSTICK_CONTROL	Selected axis is controlled by joystick
52	PI_CNTR_COMMUNICATION_ERROR	Controller detected communication error
53	PI_CNTR_DYNAMIC_MOVE_IN_PROCESS	MOV! motion still in progress
54	PI_CNTR_UNKNOWN_PARAMETER	Unknown parameter
55	PI_CNTR_NO_REP_RECORDED	No commands were recorded with REP
56	PI_CNTR_INVALID_PASSWORD	Password invalid
57	PI_CNTR_INVALID_RECORDER_CHAN	Data Record Table does not exist
58	PI_CNTR_INVALID_RECORDER_SRC_OPT	Source does not exist; number too low or too high
59	PI_CNTR_INVALID_RECORDER_SRC_CHAN	Source Record Table number too low or too high
60	PI_CNTR_PARAM_PROTECTION	Protected Param: current Command Level (CCL) too low
61	PI_CNTR_AUTOZERO_RUNNING	Command execution not possible while Autozero is running
62	PI_CNTR_NO_LINEAR_AXIS	Autozero requires at least one linear axis
63	PI_CNTR_INIT_RUNNING	Initialization still in progress
64	PI_CNTR_READ_ONLY_PARAMETER	Parameter is read-only
65	PI_CNTR_PAM_NOT_FOUND	Parameter not found in non-volatile memory
66	PI_CNTR_VOL_OUT_OF_LIMITS	Voltage out of limits
67	PI_CNTR_WAVE_TOO_LARGE	Not enough memory available for requested wave curve
68	PI_CNTR_NOT_ENOUGH_DDL_MEMORY	Not enough memory available for DDL table; DDL can not be started
69	PI_CNTR_DDL_TIME_DELAY_TOO_LARGE	Time delay larger than DDL table; DDL can not be started
70	PI_CNTR_DIFFERENT_ARRAY_LENGTH	The requested arrays have different lengths; query them separately
71	PI_CNTR_GEN_SINGLE_MODE_RESTART	Attempt to restart the generator while it is running in single step mode

72	PI_CNTR_ANALOG_TARGET_ACTIVE	Motion commands and wave generator activation are not allowed when analog target is active
73	PI_CNTR_WAVE_GENERATOR_ACTIVE	Motion commands are not allowed when wave generator is active
74	PI_CNTR_AUTOZERO_DISABLED	No sensor channel or no piezo channel connected to selected axis (sensor and piezo matrix)
75	PI_CNTR_NO_WAVE_SELECTED	Generator started (WGO) without having selected a wave table (WSL).
76	PI_CNTR_IF_BUFFER_OVERRUN	Interface buffer did overrun and command couldn't be received correctly
77	PI_CNTR_NOT_ENOUGH_RECORDED_DATA	Data Record Table does not hold enough recorded data
78	PI_CNTR_TABLE_DEACTIVATED	Data Record Table is not configured for recording
79	PI_CNTR_OPENLOOP_VALUE_SET_WHEN_SERVO_ON	Open-loop commands (SVA, SVR) are not allowed when servo is on
80	PI_CNTR_RAM_ERROR	Hardware error affecting RAM
81	PI_CNTR_MACRO_UNKNOWN_COMMAND	Not macro command
82	PI_CNTR_MACRO_PC_ERROR	Macro counter out of range
83	PI_CNTR_JOYSTICK_ACTIVE	Joystick is active
84	PI_CNTR_MOTOR_IS_OFF	Motor is off
85	PI_CNTR_ONLY_IN_MACRO	Macro-only command
86	PI_CNTR_JOYSTICK_UNKNOWN_AXIS	Invalid joystick axis
87	PI_CNTR_JOYSTICK_UNKNOWN_ID	Joystick unknown
88	PI_CNTR_REF_MODE_IS_ON	Move without referenced stage
89	PI_CNTR_NOT_ALLOWED_IN_CURRENT_MOTION_MODE	Command not allowed in current motion mode
90	PI_CNTR_DIO_AND_TRACING_NOT_POSSIBLE	No tracing possible while digital IOs are used on this HW revision. Reconnect to switch operation mode.
91	PI_CNTR_COLLISION	Move not possible, would cause collision
92	PI_CNTR_SLAVE_NOT_FAST_ENOUGH	Stage is not capable of following the master. Check the gear ratio.
93	PI_CNTR_CMD_NOT_ALLOWED_WHILE_AXIS_IN_MOTION	This command is not allowed while the affected axis or its master is in motion.
94	PI_CNTR_OPEN_LOOP_JOYSTICK_ENABLED	Servo cannot be switched on when open-loop joystick control is activated.
95	PI_CNTR_INVALID_SERVO_STATE_FOR_PARAMETER	This parameter cannot be changed in current servo mode.
96	PI_CNTR_UNKNOWN_STAGE_NAME	Unknown stage name

97	PI_CNTR_INVALID_VALUE_LENGTH	Invalid length of value (too much characters)
98	PI_CNTR_AUTOZERO_FAILED	AutoZero procedure was not successful
99	PI_CNTR_SENSOR_VOLTAGE_OFF	Sensor voltage is off
100	PI_LABVIEW_ERROR	PI driver for use with NI LabVIEW reports error. See source control for details.
200	PI_CNTR_NO_AXIS	No stage connected to axis
201	PI_CNTR_NO_AXIS_PARAM_FILE	File with axis parameters not found
202	PI_CNTR_INVALID_AXIS_PARAM_FILE	Invalid axis parameter file
203	PI_CNTR_NO_AXIS_PARAM_BACKUP	Backup file with axis parameters not found
204	PI_CNTR_RESERVED_204	PI internal error code 204
205	PI_CNTR_SMO_WITH_SERVO_ON	SMO with servo on
206	PI_CNTR_UUDECODE_INCOMPLETE_HEADER	uudecode: incomplete header
207	PI_CNTR_UUDECODE_NOTHING_TO_DECODE	uudecode: nothing to decode
208	PI_CNTR_UUDECODE_ILLEGAL_FORMAT	uudecode: illegal UUE format
209	PI_CNTR_CRC32_ERROR	CRC32 error
210	PI_CNTR_ILLEGAL_FILENAME	Illegal file name (must be 8-0 format)
211	PI_CNTR_FILE_NOT_FOUND	File not found on controller
212	PI_CNTR_FILE_WRITE_ERROR	Error writing file on controller
213	PI_CNTR_DTR_HINDERS_VELOCITY_CHANGE	VEL command not allowed in DTR Command Mode
214	PI_CNTR_POSITION_UNKNOWN	Position calculations failed
215	PI_CNTR_CONN_POSSIBLY_BROKEN	The connection between controller and stage may be broken
216	PI_CNTR_ON_LIMIT_SWITCH	The connected stage has driven into a limit switch, some controllers need CLR to resume operation
217	PI_CNTR_UNEXPECTED_STRUT_STOP	Strut test command failed because of an unexpected strut stop
218	PI_CNTR_POSITION_BASED_ON_ESTIMATION	While MOV! is running position can only be estimated!
219	PI_CNTR_POSITION_BASED_ON_INTERPOLATION	Position was calculated during MOV motion
220	PI_CNTR_INTERPOLATION_FIFO_UNDERRUN	FIFO buffer underrun during interpolation
221	PI_CNTR_INTERPOLATION_FIFO_OVERFLOW	FIFO buffer overflow during interpolation
230	PI_CNTR_INVALID_HANDLE	Invalid handle
231	PI_CNTR_NO_BIOS_FOUND	No bios found
232	PI_CNTR_SAVE_SYS_CFG_FAILED	Save system configuration failed
233	PI_CNTR_LOAD_SYS_CFG_FAILED	Load system configuration failed

301	PI_CNTR_SEND_BUFFER_OVERFLOW	Send buffer overflow
302	PI_CNTR_VOLTAGE_OUT_OF_LIMITS	Voltage out of limits
303	PI_CNTR_OPEN_LOOP_MOTION_SET_WHEN_SERVO_ON	Open-loop motion attempted when servo ON
304	PI_CNTR_RECEIVING_BUFFER_OVERFLOW	Received command is too long
305	PI_CNTR_EEPROM_ERROR	Error while reading/writing EEPROM
306	PI_CNTR_I2C_ERROR	Error on I2C bus
307	PI_CNTR_RECEIVING_TIMEOUT	Timeout while receiving command
308	PI_CNTR_TIMEOUT	A lengthy operation has not finished in the expected time
309	PI_CNTR_MACRO_OUT_OF_SPACE	Insufficient space to store macro
310	PI_CNTR_EUI_OLDVERSION_CFGDATA	Configuration data has old version number
311	PI_CNTR_EUI_INVALID_CFGDATA	Invalid configuration data
333	PI_CNTR_HARDWARE_ERROR	Internal hardware error
400	PI_CNTR_WAV_INDEX_ERROR	Wave generator index error
401	PI_CNTR_WAV_NOT_DEFINED	Wave table not defined
402	PI_CNTR_WAV_TYPE_NOT_SUPPORTED	Wave type not supported
403	PI_CNTR_WAV_LENGTH_EXCEEDS_LIMIT	Wave length exceeds limit
404	PI_CNTR_WAV_PARAMETER_NR	Wave parameter number error
405	PI_CNTR_WAV_PARAMETER_OUT_OF_LIMIT	Wave parameter out of range
406	PI_CNTR_WGO_BIT_NOT_SUPPORTED	WGO command bit not supported
500	PI_CNTR_EMERGENCY_STOP_BUTTON_ACTIVATED	The \"red knob\" is still set and disables system
501	PI_CNTR_EMERGENCY_STOP_BUTTON_WAS_ACTIVATED	The \"red knob\" was activated and still disables system - reanimation required
502	PI_CNTR_REDUNDANCY_LIMIT_EXCEEDED	Position consistency check failed
503	PI_CNTR_COLLISION_SWITCH_ACTIVATED	Hardware collision sensor(s) are activated
504	PI_CNTR_FOLLOWING_ERROR	Strut following error occurred, e.g. caused by overload or encoder failure
505	PI_CNTR_SENSOR_SIGNAL_INVALID	One sensor signal is not valid
506	PI_CNTR_SERVO_LOOP_UNSTABLE	Servo loop was unstable due to wrong parameter setting and switched off to avoid damage.
507	PI_CNTR_LOST_SPI_SLAVE_CONNECTION	Digital connection to external SPI slave device is lost
508	PI_CNTR_MOVE_ATTEMPT_NOT_PERMITTED	Move attempt not permitted due to customer or limit settings
509	PI_CNTR_TRIGGER_EMERGENCY_STOP	Emergency stop caused by trigger input
530	PI_CNTR_NODE_DOES_NOT_EXIST	A command refers to a node that does not exist

531	PI_CNTR_PARENT_NODE_DOES_NOT_EXIST	A command refers to a node that has no parent node
532	PI_CNTR_NODE_IN_USE	Attempt to delete a node that is in use
533	PI_CNTR_NODE_DEFINITION_IS_CYCLIC	Definition of a node is cyclic
536	PI_CNTR_HEXAPOD_IN_MOTION	Transformation cannot be defined as long as Hexapod is in motion
537	PI_CNTR_TRANSFORMATION_TYPE_NOT_SUPPORTED	Transformation node cannot be activated
539	PI_CNTR_NODE_PARENT_IDENTICAL_TO_CHILD	A node cannot be linked to itself
540	PI_CNTR_NODE_DEFINITION_INCONSISTENT	Node definition is erroneous or not complete (replace or delete it)
542	PI_CNTR_NODES_NOT_IN_SAME_CHAIN	The nodes are not part of the same chain
543	PI_CNTR_NODE_MEMORY_FULL	Unused nodes must be deleted before new nodes can be stored
544	PI_CNTR_PIVOT_POINT_FEATURE_NOT_SUPPORTED	With some transformations pivot point usage is not supported
545	PI_CNTR_SOFTLIMITS_INVALID	Soft limits invalid due to changes in coordinate system
546	PI_CNTR_CS_WRITE_PROTECTED	Coordinate system is write protected
547	PI_CNTR_CS_CONTENT_FROM_CONFIG_FILE	Coordinate system cannot be changed because its content is loaded from a configuration file
548	PI_CNTR_CS_CANNOT_BE_LINKED	Coordinate system may not be linked
549	PI_CNTR_KSB_CS_ROTATION_ONLY	A KSB-type coordinate system can only be rotated by multiples of 90 degrees
551	PI_CNTR_CS_DATA_CANNOT_BE_QUERIED	This query is not supported for this coordinate system type
552	PI_CNTR_CS_COMBINATION_DOES_NOT_EXIST	This combination of work-and-tool coordinate systems does not exist
553	PI_CNTR_CS_COMBINATION_INVALID	The combination must consist of one work and one tool coordinate system
554	PI_CNTR_CS_TYPE_DOES_NOT_EXIST	This coordinate system type does not exist
555	PI_CNTR_UNKNOWN_ERROR	BasMac: unknown controller error
556	PI_CNTR_CS_TYPE_NOT_ACTIVATED	No coordinate system of this type is activated
557	PI_CNTR_CS_NAME_INVALID	Name of coordinate system is invalid
558	PI_CNTR_CS_GENERAL_FILE_MISSING	File with stored CS systems is missing or erroneous
559	PI_CNTR_CS_LEVELING_FILE_MISSING	File with leveling CS is missing or erroneous
601	PI_CNTR_NOT_ENOUGH_MEMORY	not enough memory
602	PI_CNTR_HW_VOLTAGE_ERROR	hardware voltage error

603	PI_CNTR_HW_TEMPERATURE_ERROR	hardware temperature out of range
604	PI_CNTR_POSITION_ERROR_TOO_HIGH	Position error of any axis in the system is too high
606	PI_CNTR_INPUT_OUT_OF_RANGE	Maximum value of input signal has been exceeded
607	PI_CNTR_NO_INTEGER	Value is not integer
608	PI_CNTR_FAST_ALIGNMENT_PROCESS_IS_NOT_RUNNING	Fast alignment process cannot be paused because it is not running
609	PI_CNTR_FAST_ALIGNMENT_PROCESS_IS_NOT_PAUSED	Fast alignment process cannot be restarted/resumed because it is not paused
650	PI_CNTR_UNABLE_TO_SET_PARAM_WITH_SPA	Parameter could not be set with SPA - SEP needed?
651	PI_CNTR_PHASE_FINDING_ERROR	Phase finding error
652	PI_CNTR_SENSOR_SETUP_ERROR	Sensor setup error
653	PI_CNTR_SENSOR_COMM_ERROR	Sensor communication error
654	PI_CNTR_MOTOR_AMPLIFIER_ERROR	Motor amplifier error
655	PI_CNTR_OVER_CURR_PROTEC_TRIGGERED_BY_I2T	Overcurrent protection triggered by I2T-module
656	PI_CNTR_OVER_CURR_PROTEC_TRIGGERED_BY_AMP_MODULE	Overcurrent protection triggered by amplifier module
657	PI_CNTR_SAFETY_STOP_TRIGGERED	Safety stop triggered
658	PI_SENSOR_OFF	Sensor off?
659	PI_CNTR_PARAM_CONFLICT	Parameter could not be set. Conflict with another parameter.
700	PI_CNTR_COMMAND_NOT_ALLOWED_IN_EXTERNAL_MODE	Command not allowed in external mode
710	PI_CNTR_EXTERNAL_MODE_ERROR	External mode communication error
715	PI_CNTR_INVALID_MODE_OF_OPERATION	Invalid mode of operation
716	PI_CNTR_FIRMWARE_STOPPED_BY_CMD	Firmware stopped by command (#27)
717	PI_CNTR_EXTERNAL_MODE_DRIVER_MISSING	External mode driver missing
718	PI_CNTR_CONFIGURATION_FAILURE_EXTERNAL_MODE	Missing or incorrect configuration of external mode
719	PI_CNTR_EXTERNAL_MODE_CYCLETIME_INVALID	External mode cycletime invalid
720	PI_CNTR_BRAKE_ACTIVATED	Brake is activated
725	PI_CNTR_DRIVE_STATE_TRANSITION_ERROR	Drive state transition error
731	PI_CNTR_SURFACEDETECTION_RUNNING	Command not allowed while surface detection is running
732	PI_CNTR_SURFACEDETECTION_FAILED	Last surface detection failed
733	PI_CNTR_FIELDBUS_IS_ACTIVE	Fieldbus is active and is blocking GCS control commands
1000	PI_CNTR_TOO_MANY_NESTED_MACROS	Too many nested macros
1001	PI_CNTR_MACRO_ALREADY_DEFINED	Macro already defined
1002	PI_CNTR_NO_MACRO_RECORDING	Macro recording not activated

1003	PI_CNTR_INVALID_MAC_PARAM	Invalid parameter for MAC
1004	PI_CNTR_RESERVED_1004	PI internal error code 1004
1005	PI_CNTR_CONTROLLER_BUSY	Controller is busy with some lengthy operation (e.g. reference move, fast scan algorithm)
1006	PI_CNTR_INVALID_IDENTIFIER	Invalid identifier (invalid special characters, ...)
1007	PI_CNTR_UNKNOWN_VARIABLE_OR_ARGUMENT	Variable or argument not defined
1008	PI_CNTR_RUNNING_MACRO	Controller is (already) running a macro
1009	PI_CNTR_MACRO_INVALID_OPERATOR	Invalid or missing operator for condition. Check necessary spaces around operator.
1010	PI_CNTR_MACRO_NO_ANSWER	No response was received while executing WAC/MEX/JRC/...
1011	PI_CMD_NOT_VALID_IN_MACRO_MODE	Command not valid during macro execution
1012	PI_CNTR_ERROR_IN_MACRO	Error occured during macro execution
1024	PI_CNTR_MOTION_ERROR	Motion error: position error too large, servo is switched off automatically
1025	PI_CNTR_MAX_MOTOR_OUTPUT_REACHED	Maximum motor output reached
1063	PI_CNTR_EXT_PROFILE_UNALLOWED_CMD	User Profile Mode: Command is not allowed, check for required preparatory commands
1064	PI_CNTR_EXT_PROFILE_EXPECTING_MOTION_ERROR	User Profile Mode: First target position in User Profile is too far from current position
1065	PI_CNTR_PROFILE_ACTIVE	Controller is (already) in User Profile Mode
1066	PI_CNTR_PROFILE_INDEX_OUT_OF_RANGE	User Profile Mode: Block or Data Set index out of allowed range
1071	PI_CNTR_PROFILE_OUT_OF_MEMORY	User Profile Mode: Out of memory
1072	PI_CNTR_PROFILE_WRONG_CLUSTER	User Profile Mode: Cluster is not assigned to this axis
1073	PI_CNTR_PROFILE_UNKNOWN_CLUSTER_IDENTIFIER	Unknown cluster identifier
1090	PI_CNTR_TOO_MANY_TCP_CONNECTIONS_OPEN	There are too many open tcpip connections
2000	PI_CNTR_ALREADY_HAS_SERIAL_NUMBER	Controller already has a serial number
4000	PI_CNTR_SECTOR_ERASE_FAILED	Sector erase failed
4001	PI_CNTR_FLASH_PROGRAM_FAILED	Flash program failed
4002	PI_CNTR_FLASH_READ_FAILED	Flash read failed
4003	PI_CNTR_HW_MATCHCODE_ERROR	HW match code missing/invalid
4004	PI_CNTR_FW_MATCHCODE_ERROR	FW match code missing/invalid

4005	PI_CNTR_HW_VERSION_ERROR	HW version missing/invalid
4006	PI_CNTR_FW_VERSION_ERROR	FW version missing/invalid
4007	PI_CNTR_FW_UPDATE_ERROR	FW update failed
4008	PI_CNTR_FW_CRC_PAR_ERROR	FW Parameter CRC wrong
4009	PI_CNTR_FW_CRC_FW_ERROR	FW CRC wrong
5000	PI_CNTR_INVALID_PCC_SCAN_DATA	PicoCompensation scan data is not valid
5001	PI_CNTR_PCC_SCAN_RUNNING	PicoCompensation is running, some actions can not be executed during scanning/recording
5002	PI_CNTR_INVALID_PCC_AXIS	Specified axis cannot be defined as PPC axis
5003	PI_CNTR_PCC_SCAN_OUT_OF_RANGE	Defined scan area is larger than the travel range
5004	PI_CNTR_PCC_TYPE_NOT_EXISTING	Specified PicoCompensation type is not defined
5005	PI_CNTR_PCC_PAM_ERROR	PicoCompensation parameter error
5006	PI_CNTR_PCC_TABLE_ARRAY_TOO_LARGE	PicoCompensation table is larger than maximum table length
5100	PI_CNTR_NEXLINE_ERROR	Common error in NEXLINE® firmware module
5101	PI_CNTR_CHANNEL_ALREADY_USED	Output channel for NEXLINE® can not be redefined for other usage
5102	PI_CNTR_NEXLINE_TABLE_TOO_SMALL	Memory for NEXLINE® signals is too small
5103	PI_CNTR_RNP_WITH_SERVO_ON	RNP can not be executed if axis is in closed loop
5104	PI_CNTR_RNP_NEEDED	Relax procedure (RNP) needed
5200	PI_CNTR_AXIS_NOT_CONFIGURED	Axis must be configured for this action
5300	PI_CNTR_FREQU_ANALYSIS_FAILED	Frequency analysis failed
5301	PI_CNTR_FREQU_ANALYSIS_RUNNING	Another frequency analysis is running
6000	PI_CNTR_SENSOR_ABS_INVALID_VALUE	Invalid preset value of absolute sensor
6001	PI_CNTR_SENSOR_ABS_WRITE_ERROR	Error while writing to sensor
6002	PI_CNTR_SENSOR_ABS_READ_ERROR	Error while reading from sensor
6003	PI_CNTR_SENSOR_ABS_CRC_ERROR	Checksum error of absolute sensor
6004	PI_CNTR_SENSOR_ABS_ERROR	General error of absolute sensor
6005	PI_CNTR_SENSOR_ABS_OVERFLOW	Overflow of absolute sensor position

Interface errors

0	COM_NO_ERROR	No error occurred during function call
-1	COM_ERROR	Error during com operation (could not be specified)

-2	SEND_ERROR	Error while sending data
-3	REC_ERROR	Error while receiving data
-4	NOT_CONNECTED_ERROR	Not connected (no port with specified ID open)
-5	COM_BUFFER_OVERFLOW	Buffer overflow
-6	CONNECTION_FAILED	Error while opening port
-7	COM_TIMEOUT	Timeout error
-8	COM_MULTILINE_RESPONSE	There are more lines waiting in buffer
-9	COM_INVALID_ID	There is no interface or DLL handle with the specified ID
-10	COM_NOTIFY_EVENT_ERROR	Event/message for notification could not be opened
-11	COM_NOT_IMPLEMENTED	Function not supported by this interface type
-12	COM_ECHO_ERROR	Error while sending "echoed" data
-13	COM_GPIB_EDVR	IEEE488: System error
-14	COM_GPIB_ECIC	IEEE488: Function requires GPIB board to be CIC
-15	COM_GPIB_ENOL	IEEE488: Write function detected no listeners
-16	COM_GPIB_EADR	IEEE488: Interface board not addressed correctly
-17	COM_GPIB_EARG	IEEE488: Invalid argument to function call
-18	COM_GPIB_ESAC	IEEE488: Function requires GPIB board to be SAC
-19	COM_GPIB_EABO	IEEE488: I/O operation aborted
-20	COM_GPIB_ENEB	IEEE488: Interface board not found
-21	COM_GPIB_EDMA	IEEE488: Error performing DMA
-22	COM_GPIB_EOIP	IEEE488: I/O operation started before previous operation completed
-23	COM_GPIB_ECAP	IEEE488: No capability for intended operation
-24	COM_GPIB_EFSO	IEEE488: File system operation error
-25	COM_GPIB_EBUS	IEEE488: Command error during device call
-26	COM_GPIB_ESTB	IEEE488: Serial poll-status byte lost
-27	COM_GPIB_ESRQ	IEEE488: SRQ remains asserted
-28	COM_GPIB_ETAB	IEEE488: Return buffer full
-29	COM_GPIB_ELCK	IEEE488: Address or board locked
-30	COM_RS_INVALID_DATA_BITS	RS-232: 5 data bits with 2 stop bits is an invalid combination, as is 6, 7, or 8 data bits with 1.5 stop bits
-31	COM_ERROR_RS_SETTINGS	RS-232: Error configuring the COM port

-32	COM_INTERNAL_RESOURCES_ERROR	Error dealing with internal system resources (events, threads, ...)
-33	COM_DLL_FUNC_ERROR	A DLL or one of the required functions could not be loaded
-34	COM_FTDIUSB_INVALID_HANDLE	FTDIUSB: invalid handle
-35	COM_FTDIUSB_DEVICE_NOT_FOUND	FTDIUSB: device not found
-36	COM_FTDIUSB_DEVICE_NOT_OPENED	FTDIUSB: device not opened
-37	COM_FTDIUSB_IO_ERROR	FTDIUSB: IO error
-38	COM_FTDIUSB_INSUFFICIENT_RESOURCES	FTDIUSB: insufficient resources
-39	COM_FTDIUSB_INVALID_PARAMETER	FTDIUSB: invalid parameter
-40	COM_FTDIUSB_INVALID_BAUD_RATE	FTDIUSB: invalid baud rate
-41	COM_FTDIUSB_DEVICE_NOT_OPENED_FOR_ERASE	FTDIUSB: device not opened for erase
-42	COM_FTDIUSB_DEVICE_NOT_OPENED_FOR_WRITE	FTDIUSB: device not opened for write
-43	COM_FTDIUSB_FAILED_TO_WRITE_DEVICE	FTDIUSB: failed to write device
-44	COM_FTDIUSB_EEPROM_READ_FAILED	FTDIUSB: EEPROM read failed
-45	COM_FTDIUSB_EEPROM_WRITE_FAILED	FTDIUSB: EEPROM write failed
-46	COM_FTDIUSB_EEPROM_ERASE_FAILED	FTDIUSB: EEPROM erase failed
-47	COM_FTDIUSB_EEPROM_NOT_PRESENT	FTDIUSB: EEPROM not present
-48	COM_FTDIUSB_EEPROM_NOT_PROGRAMMED	FTDIUSB: EEPROM not programmed
-49	COM_FTDIUSB_INVALID_ARGS	FTDIUSB: invalid arguments
-50	COM_FTDIUSB_NOT_SUPPORTED	FTDIUSB: not supported
-51	COM_FTDIUSB_OTHER_ERROR	FTDIUSB: other error
-52	COM_PORT_ALREADY_OPEN	Error while opening the COM port: was already open
-53	COM_PORT_CHECKSUM_ERROR	Checksum error in received data from COM port
-54	COM_SOCKET_NOT_READY	Socket not ready, you should call the function again
-55	COM_SOCKET_PORT_IN_USE	Port is used by another socket
-56	COM_SOCKET_NOT_CONNECTED	Socket not connected (or not valid)
-57	COM_SOCKET_TERMINATED	Connection terminated (by peer)
-58	COM_SOCKET_NO_RESPONSE	Can't connect to peer
-59	COM_SOCKET_INTERRUPTED	Operation was interrupted by a nonblocked signal
-60	COM_PCI_INVALID_ID	No device with this ID is present
-61	COM_PCI_ACCESS_DENIED	Driver could not be opened (on Vista: run as administrator!)
-62	COM_SOCKET_HOST_NOT_FOUND	Host not found
-63	COM_DEVICE_CONNECTED	Device already connected

DLL errors

-1001	PI_UNKNOWN_AXIS_IDENTIFIER	Unknown axis identifier

-1002	PI_NR_NAV_OUT_OF_RANGE	Number for NAV out of range--must be in [1.10000]
-1003	PI_INVALID_SGA	Invalid value for SGA--must be one of 1, 10, 100, 1000
-1004	PI_UNEXPECTED_RESPONSE	Controller sent unexpected response
-1005	PI_NO_MANUAL_PAD	No manual control pad installed, calls to SMA and related commands are not allowed
-1006	PI_INVALID_MANUAL_PAD_KNOB	Invalid number for manual control pad knob
-1007	PI_INVALID_MANUAL_PAD_AXIS	Axis not currently controlled by a manual control pad
-1008	PI_CONTROLLER_BUSY	Controller is busy with some lengthy operation (e.g., reference move, fast scan algorithm)
-1009	PI_THREAD_ERROR	Internal error--could not start thread
-1010	PI_IN_MACRO_MODE	Controller is (already) in macro mode--command not valid in macro mode
-1011	PI_NOT_IN_MACRO_MODE	Controller not in macro mode--command not valid unless macro mode active
-1012	PI_MACRO_FILE_ERROR	Could not open file to write or read macro
-1013	PI_NO_MACRO_OR_EMPTY	No macro with specified name on controller, or macro is empty
-1014	PI_MACRO_EDITOR_ERROR	Internal error in macro editor
-1015	PI_INVALID_ARGUMENT	One or more arguments specified for function invalid (empty string, index out of range, ...)
-1016	PI_AXIS_ALREADY_EXISTS	Axis identifier is already in use by a connected stage
-1017	PI_INVALID_AXIS_IDENTIFIER	Invalid axis identifier
-1018	PI_COM_ARRAY_ERROR	Could not access array data in COM server
-1019	PI_COM_ARRAY_RANGE_ERROR	Range of array does not fit the number of parameters
-1020	PI_INVALID_SPA_CMD_ID	Invalid parameter ID specified for SPA or SPA?
-1021	PI_NR_AVG_OUT_OF_RANGE	Number for AVG out of range--must be >0
-1022	PI_WAV_SAMPLES_OUT_OF_RANGE	Incorrect number of samples specified for WAV
-1023	PI_WAV_FAILED	Generation of wave failed
-1024	PI_MOTION_ERROR	Motion error: position error too large, servo is switched off automatically

-1025	PI_RUNNING_MACRO	Controller is (already) running a macro
-1026	PI_PZT_CONFIG_FAILED	Configuration of PZT stage or amplifier failed
-1027	PI_PZT_CONFIG_INVALID_PARAMS	Current settings are not valid for desired configuration
-1028	PI_UNKNOWN_CHANNEL_IDENTIFIER	Unknown channel identifier
-1029	PI_WAVE_PARAM_FILE_ERROR	Error while reading/writing wave generator parameter file
-1030	PI_UNKNOWN_WAVE_SET	Could not find description of wave form. Maybe WG.INI is missing?
-1031	PI_WAVE_EDITOR_FUNC_NOT_LOADED	The WGWaveEditor DLL function was not found at startup
-1032	PI_USER_CANCELLED	The user cancelled a dialog
-1033	PI_C844_ERROR	Error from C-844 Controller
-1034	PI_DLL_NOT_LOADED	DLL necessary to call function not loaded, or function not found in DLL
-1035	PI_PARAMETER_FILE_PROTECTED	The open parameter file is protected and cannot be edited
-1036	PI_NO_PARAMETER_FILE_OPENED	There is no parameter file open
-1037	PI_STAGE_DOES_NOT_EXIST	Selected stage does not exist
-1038	PI_PARAMETER_FILE_ALREADY_OPENED	There is already a parameter file open. Close it before opening a new file
-1039	PI_PARAMETER_FILE_OPEN_ERROR	Could not open parameter file
-1040	PI_INVALID_CONTROLLER_VERSION	The version of the connected controller is invalid
-1041	PI_PARAM_SET_ERROR	Parameter could not be set with SPA--parameter not defined for this controller!
-1042	PI_NUMBER_OF_POSSIBLE_WAVES_EXCEEDED	The maximum number of wave definitions has been exceeded
-1043	PI_NUMBER_OF_POSSIBLE_GENERATORS_EXCEEDED	The maximum number of wave generators has been exceeded
-1044	PI_NO_WAVE_FOR_AXIS_DEFINED	No wave defined for specified axis
-1045	PI_CANT_STOP_OR_START_WAV	Wave output to axis already stopped/started
-1046	PI_REFERENCE_ERROR	Not all axes could be referenced
-1047	PI_REQUIRED_WAVE_NOT_FOUND	Could not find parameter set required by frequency relation
-1048	PI_INVALID_SPP_CMD_ID	Command ID specified for SPP or SPP? is not valid
-1049	PI_STAGE_NAME_ISNT_UNIQUE	A stage name specified for CST is not unique

-1050	PI_FILE_TRANSFER_BEGIN_MISSING	A uuencoded file transferred did not start with "begin" followed by the proper filename
-1051	PI_FILE_TRANSFER_ERROR_TEMP_FILE	Could not create/read file on host PC
-1052	PI_FILE_TRANSFER_CRC_ERROR	Checksum error when transferring a file to/from the controller
-1053	PI_COULDNT_FIND_PISTAGES_DAT	The PiStages.dat database could not be found. This file is required to connect a stage with the CST command
-1054	PI_NO_WAVE_RUNNING	No wave being output to specified axis
-1055	PI_INVALID_PASSWORD	Invalid password
-1056	PI_OPM_COM_ERROR	Error during communication with OPM (Optical Power Meter), maybe no OPM connected
-1057	PI_WAVE_EDITOR_WRONG_PARAMNUM	WaveEditor: Error during wave creation, incorrect number of parameters
-1058	PI_WAVE_EDITOR_FREQUENCY_OUT_OF_RANGE	WaveEditor: Frequency out of range
-1059	PI_WAVE_EDITOR_WRONG_IP_VALUE	WaveEditor: Error during wave creation, incorrect index for integer parameter
-1060	PI_WAVE_EDITOR_WRONG_DP_VALUE	WaveEditor: Error during wave creation, incorrect index for floating point parameter
-1061	PI_WAVE_EDITOR_WRONG_ITEM_VALUE	WaveEditor: Error during wave creation, could not calculate value
-1062	PI_WAVE_EDITOR_MISSING_GRAPH_COMPONENT	WaveEditor: Graph display component not installed
-1063	PI_EXT_PROFILE_UNALLOWED_CMD	User profile mode: command is not allowed, check for required preparatory commands
-1064	PI_EXT_PROFILE_EXPECTING_MOTION_ERROR	User profile mode: first target position in user profile is too far from current position
-1065	PI_EXT_PROFILE_ACTIVE	Controller is (already) in user profile mode
-1066	PI_EXT_PROFILE_INDEX_OUT_OF_RANGE	User profile mode: block or data set index out of allowed range
-1067	PI_PROFILE_GENERATOR_NO_PROFILE	ProfileGenerator: No profile has been created yet
-1068	PI_PROFILE_GENERATOR_OUT_OF_LIMITS	ProfileGenerator: Generated profile exceeds limits of one or both axes
-1069	PI_PROFILE_GENERATOR_UNKNOWN_PARAMETER	ProfileGenerator: Unknown parameter ID in Set/Get Parameter command

-1070	PI_PROFILE_GENERATOR_PAR_OUT_OF_RANGE	ProfileGenerator: Parameter out of allowed range
-1071	PI_EXT_PROFILE_OUT_OF_MEMORY	User profile mode: out of memory
-1072	PI_EXT_PROFILE_WRONG_CLUSTER	User profile mode: cluster is not assigned to this axis
-1073	PI_UNKNOWN_CLUSTER_IDENTIFIER	Unknown cluster identifier
-1074	PI_INVALID_DEVICE_DRIVER_VERSION	The installed device driver doesn't match the required version. Please refer to the documentation to determine the required device driver version.
-1075	PI_INVALID_LIBRARY_VERSION	The library used doesn't match the required version. Please refer to the documentation to determine the required library version.
-1076	PI_INTERFACE_LOCKED	The interface is currently locked by another function. Please try again later.
-1077	PI_PARAM_DAT_FILE_INVALID_VERSION	Version of parameter DAT file does not match the required version. Current files are available at www.pi.ws.
-1078	PI_CANNOT_WRITE_TO_PARAM_DAT_FILE	Cannot write to parameter DAT file to store user defined stage type.
-1079	PI_CANNOT_CREATE_PARAM_DAT_FILE	Cannot create parameter DAT file to store user defined stage type.
-1080	PI_PARAM_DAT_FILE_INVALID_REVISION	Parameter DAT file does not have correct revision.
-1081	PI_USERSTAGES_DAT_FILE_INVALID_REVISION	User stages DAT file does not have correct revision.
-1082	PI_SOFTWARE_TIMEOUT	Timeout Error. Some lengthy operation did not finish within expected time.
-1083	PI_WRONG_DATA_TYPE	A function argument has an unexpected data type.
-1084	PI_DIFFERENT_ARRAY_SIZES	Length of data arrays is different.
-1085	PI_PARAM_NOT_FOUND_IN_PARAM_DAT_FILE	Parameter value not found in parameter DAT file.
-1086	PI_MACRO_RECORDING_NOT_ALLOWED_IN_THIS_MODE	Macro recording is not allowed in this mode of operation.
-1087	PI_USER_CANCELLED_COMMAND	Command cancelled by user input.
-1088	PI_TOO_FEW_GCS_DATA	Controller sent too few GCS data sets
-1089	PI_TOO_MANY_GCS_DATA	Controller sent too many GCS data sets
-1090	PI_GCS_DATA_READ_ERROR	Communication error while reading GCS data
-1091	PI_WRONG_NUMBER_OF_INPUT_ARGUMENTS	Wrong number of input arguments.

-1092	PI_FAILED_TO_CHANGE_CCL_LEVEL	Change of command level has failed.
-1093	PI_FAILED_TO_SWITCH_OFF_SERVO	Servo mode has failed to switch off.
-1094	PI_FAILED_TO_SET_SINGLE_PARAMETER_WHILE_PERFORMING_CST	A parameter could not be set while performing CST: CST was not performed (parameters remain unchanged).
-1095	PI_ERROR_CONTROLLER_REBOOT	Connection could not be reestablished after reboot.
-1096	PI_ERROR_AT_QHPA	Sending HPA? or receiving the response has failed.
-1097	PI_QHPA_NONCOMPLIANT_WITH_GCS	HPA? response does not comply with GCS2 syntax.
-1098	PI_FAILED_TO_READ_QSPA	Response to SPA? could not be received. Response to SPA? could not be received.
-1099	PI_PAM_FILE_WRONG_VERSION	Version of PAM file cannot be handled (too old or too new)
-1100	PI_PAM_FILE_INVALID_FORMAT	PAM file does not contain required data in PAM-file format
-1101	PI_INCOMPLETE_INFORMATION	Information does not contain all required data
-1102	PI_NO_VALUE_AVAILABLE	No value for parameter available
-1103	PI_NO_PAM_FILE_OPEN	No PAM file is open
-1104	PI_INVALID_VALUE	Invalid value
-1105	PI_UNKNOWN_PARAMETER	Unknown parameter
-1106	PI_RESPONSE_TO_QSEP_FAILED	Response to SEP? could not be received.
-1107	PI_RESPONSE_TO_QSPA_FAILED	Response to SPA? could not be received.
-1108	PI_ERROR_IN_CST_VALIDATION	Error while performing CST: One or more parameters were not set correctly.
-1109	PI_ERROR_PAM_FILE_HAS_DUPLICATE_ENTRY_WITH_DIFFERENT_VALUES	PAM file has duplicate entry with different values.
-1110	PI_ERROR_FILE_NO_SIGNATURE	File has no signature
-1111	PI_ERROR_FILE_INVALID_SIGNATURE	File has invalid signature
-10000	PI_PARAMETER_DB_INVALID_STAGE_TYPE_FORMAT	PI stage database: String containing stage type and description has invalid format.
-10001	PI_PARAMETER_DB_SYSTEM_NOT_AVAILABLE	PI stage database: Database does not contain the selected stage type for the connected controller.
-10002	PI_PARAMETER_DB_FAILED_TO_ESTABLISH_CONNECTION	PI stage database: Establishing the connection has failed.
-10003	PI_PARAMETER_DB_COMMUNICATION_ERROR	PI stage database: Communication was interrupted (e.g. because database was deleted).

-10004	PI_PARAMETER_DB_ERROR_WHILE_QUERYING_PARAMETERS	PI stage database: Querying data failed.
-10005	PI_PARAMETER_DB_SYSTEM_ALREADY_EXISTS	PI stage database: System already exists. Rename stage and try again.
-10006	PI_PARAMETER_DB_QHPA_CONTANS_UNKNOWN_PAM_IDS	PI stage database: Response to HPA? contains unknown parameter IDs.
-10007	PI_PARAMETER_DB_AND_QHPA_ARE_INCONSISTENT	PI stage database: Inconsistency between database and response to HPA?.
-10008	PI_PARAMETER_DB_SYSTEM_COULD_NOT_BE_ADDED	PI stage database: Stage has not been added.
-10009	PI_PARAMETER_DB_SYSTEM_COULD_NOT_BE_REMOVED	PI stage database: Stage has not been removed.
-10010	PI_PARAMETER_DB_CONTROLLER_DB_PARAMETERS_MISMATCH	Controller does not support all stage parameters stored in PI stage database. No parameters were set.
-10011	PI_PARAMETER_DB_DATABASE_IS_OUTDATED	The version of PISTAGES3.DB stage database is out of date. Please update via PIUpdateFinder. No parameters were set.
-10012	PI_PARAMETER_DB_AND_HPA_MISMATCH_STRICT	Mismatch between number of parameters present in stage database and available in controller interface. No parameters were set.
-10013	PI_PARAMETER_DB_AND_HPA_MISMATCH_LOOSE	Mismatch between number of parameters present in stage database and available in controller interface. Some parameters were ignored.
-10014	PI_PARAMETER_DB_FAILED_TO_SET_PARAMETERS_CORRECTLY	One or more parameters could not be set correctly on the controller.
-10015	PI_PARAMETER_DB_MISSING_PARAMETER_DEFINITIONS_IN_DATABASE	One or more parameter definitions are not present in stage database. Please update PISTAGES3.DB via PIUpdateFinder. Missing parameters were ignored.

9 Adapting Settings

In this Chapter

9.1 Overview of the C-887's Settings

Various settings can be stored as default values for the C-887 in the nonvolatile memory so that they are preserved when the C-887 is switched off or rebooted. The default values for the settings come from different sources:

- Configuration files: see "Updating Firmware and Configuration Files" (p. 318)
- Positioner databases: Refer to "Operating Parameters for Axes A and B" (p. 30)
- Interface parameters: Adapting in the nonvolatile memory is possible with the `IFS` command (p. 196), further information in "Establishing Communication via the TCP/IP Interface" (p. 72) and "Establishing Communication via the RS-232 Interface" (p. 78)
- Commands for working with user-defined coordinate systems: Further information in "Coordinate Systems" (p. 38)
- Parameters that are changed with the `SPA` command (p. 244) and saved with the `WPA` command (p. 276). These parameters can be divided into the following categories:
  - Protected parameters whose default settings cannot be changed
  - Parameters that can be set by the user to adapt to the application

  Write permission for the parameters is determined by command levels (p. 151).

  Further information in "Changing Parameter Values in the C-887" (p. 299).

9.2 Changing Parameter Values in the C-887

Each parameter is in the C-887's volatile and nonvolatile memory. The values in the nonvolatile memory are loaded to the volatile memory as default values when switching on or rebooting the C-887. The values in the volatile memory determine the current behavior of the system.

Determining available parameters

- Send the HPA? command (p. 193) to obtain a list and a short description of all parameters available for the C-887.

Changing parameter values in the volatile memory

INFORMATION

Changing parameter values can cause undesirable results.

- Create a backup copy on the PC before changing the parameter settings of the C-887; see "Saving Parameter Values in a Text File" (p. 301). You can then restore the original settings at any time.

- Send the SPA? command (p. 245) to obtain a list of parameter values in the volatile memory of the C-887.
- Change the parameter values in the volatile memory:
  a) If write access to the parameter values is necessary, send the `CCL 1 advanced` command to go to command level 1.
  b) Change the parameter values in the volatile memory of the C-887 with the SPA command (p. 244).

INFORMATION

Parameter values are also changed by the CST command (p. 153), which assigns a positioner type to axes A and B. This loads the corresponding parameter values from a positioner database into the volatile memory. You can find further information in "Operating Parameters for Axes A and B" (p. 30).

Writing parameter values from the volatile memory to the nonvolatile memory

INFORMATION

To save parameter values in the nonvolatile memory with the WPA command, it is necessary to enter a password. Usable passwords:

100	Saves the currently valid values of all parameters, the currently valid settings for coordinate systems, and the current assignment of positioner types to axes A and B
101	Saves the currently valid values of all parameters. Parameters can be selected individually.
SKS	Saves the currently valid settings for coordinate systems
A12	Assignment of positioner types to axes A and B:

- Write the current parameter values to the nonvolatile memory of the C-887 with the `WPA` command (p. 276).

  Alternative procedure if you are working with PIMikroMove:

  a) Select the ***C-887 > Save parameters to nonvolatile memory*** menu item in the main window of PIMikroMove. The ***Save Parameters to Non-Volatile Memory*** dialog opens.

  b) Enter either the corresponding password into the ***Save Parameters to Non-Volatile Memory*** dialog or select the corresponding entry.

  c) Click ***OK*** to save and to close the dialog.

9.3 Saving Parameter Values in a Text File

Requirements

- ✓ You have read and understood the General Notes on Startup (p. 67).
- ✓ You have established communication between the C-887 and the PC with PIMikroMove or PITerminal via TCP/IP (p. 76) or RS-232 (p. 78).

Saving parameter values in a text file

1. If you use PIMikroMove, open the window for sending commands:

   – Select the ***Tools > Command entry*** menu item in the main window or press F4 on the keyboard.

   In PITerminal, the main window from which commands can be sent is opened automatically after establishing communication.

2. Get the current parameter values of the C-887 with the `SPA?` command.

3. Click on the ***Save...*** button.

   The ***Save content of terminal as textfile*** window opens.

4. Save the queried parameter values to a text file on your PC in the ***Save content of terminal as textfile*** window.

9.4 Parameter Overview

INFORMATION

The write access for the parameters of the C-887 is defined by command levels. After the controller is switched on or rebooted, the active command level is always level 0. For particular parameters, write access is only allowed on command level 1. On command levels > 1, write access is only available to PI service personnel.

- If necessary, send the `CCL 1 advanced` command or enter the password `advanced` to change to command level 1.
- Contact the customer service department if there seem to be problems with parameters of command level 2 or higher (p. 343).

Parameter ID (hexa decimal)	Command level for write access	Affected element type	Data type	Parameter name (Unit)	Description
0x1	0	Hexapod struts 1 to 6, axes A and B	INT	P term	Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x2	0	Hexapod struts 1 to 6, axes A and B	INT	I term	Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x3	0	Hexapod struts 1 to 6, axes A and B	INT	D term	Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x4	0	Hexapod struts 1 to 6, axes A and B	INT	I limit	Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x5	0	Hexapod struts 1 to 6, axes A and B	INT	Velocity Feed Forward	Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x8	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Maximum Position Error (mm)	Maximum position error Is used for detecting motion errors. If the position deviation (i.e., the absolute value of the difference between the current position and the commanded position) exceeds the specified maximum value in closed-loop operation, the C-887 sets the error code +1024 ("Motion error"). For details on error handling, see "Automatically Switching off the Servo Mode / Stopping the Motion" (p. 88). Is changed in the volatile memory for axes A and B when a positioner type is assigned.

Parameter ID (hexa decimal)	Command level for write access	Affected element type	Data type	Parameter name (Unit)	Description
0x9	0	Hexapod struts 1 to 6, axes A and B	INT	Maximum Motor Output	Maximum permissible absolute measure of the control value of the motor driver (dimensionless) 0 to 32767 Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0xA	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Maximum Allowed Velocity	Maximum velocity Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0xB	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Current Accelera-tion	Acceleration Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0xC	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Current Decelera-tion	Delay Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0xE	0	Hexapod struts 1 to 6, axes A and B	INT	Numerator Of The Counts-Per-Physical-Unit Factor	Numerator and denominator of the factor for counts per physical length unit 1 to 1,000,000 for each parameter. The factor for counts per physical length unit determines the unit for motion commands. Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0xF	0	Hexapod struts 1 to 6, axes A and B	INT	Denomina-tor Of The Counts-Per-Physical-Unit Factor	
0x10	1	Hexapod struts 1 to 6, axes A and B	INT	Output Mode	1 = PWM Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x13	0	Hexapod struts 1 to 6, axes A and B	INT	Is Rotary Stage?	Is this a rotation stage? 0 = Not a rotation stage 1 = Rotation stage No evaluation by the C-887, but only by the PC software: PIMikroMove determines which motion is permissible on the basis of this value. Is changed in the volatile memory for axes A and B when a positioner type is assigned.

Parameter ID (hexa decimal)	Command level for write access	Affected element type	Data type	Parameter name (Unit)	Description
0x14	0	Hexapod struts 1 to 6, axes A and B	INT	Stage has reference switch	Is a reference switch available? 0 = Reference switch not installed 1 = Reference switch available Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x15	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Maximum Travel In Positive Direction (Phys. Unit)	Soft limit in positive direction Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x16	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Value at reference switch position	Position value at the reference switch Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x17	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Distance between Negative Limit To Reference Position (Phys. Unit)	Distance between the reference switch and the negative limit switch Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x18	0	Hexapod struts 1 to 6, axes A and B	INT	Limit Mode	Signal logic of the limit switches Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x1A	0	Hexapod struts 1 to 6, axes A and B	INT	Has brake?	Is a brake available? 0 = Brake not available 1 = Brake available Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x2F	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Distance between Reference Position To Positive Limit (Phys. Unit)	Distance between reference switch and positive limit switch Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x30	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Maximum Travel In Negative Direction (Phys. Unit)	Soft limit in a negative direction Is changed in the volatile memory for axes A and B when a positioner type is assigned.

Parameter ID (hexa decimal)	Command level for write access	Affected element type	Data type	Parameter name (Unit)	Description
0x31	0	Hexapod struts 1 to 6, axes A and B	INT	Invert Reference Switch Signal	Should the reference signal be inverted? Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x32	0	Hexapod struts 1 to 6, axes A and B	INT	Has No Limit Switches	Are limit switches available? 0 = Limit switch evaluation activated. Only effective when the mechanics actually supply the limit switch signals to the C-887. 1 = Limit switch evaluation deactivated. Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x33	1	Hexapod struts 1 to 6, axes A and B	INT	Offset for the positive direction of motion	Offset for the positive direction of motion Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x34	1	Hexapod struts 1 to 6, axes A and B	INT	Offset for the negative direction of motion	Offset for the negative direction of motion Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x36	0	Hexapod struts 1 to 6, axes A and B	INT	Settling Window (encoder counts)	Settling window around the target position Used for determining the motion status (p. 42). Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x38	0	Hexapod struts 1 to 6, axes A and B	INT	Settle Time	Delay time Specified in servo cycles Used for determining the motion status (p. 42).
0x3B	0	Hexapod struts 1 to 6, axes A and B	INT	Accelera-tion feed forward	
0x3C	0	Hexapod struts 1 to 6, axes A and B	CHAR	Stage Name	Positioner name Is changed in the volatile memory for axes A and B when a positioner type is assigned.

Parameter ID (hexa decimal)	Command level for write access	Affected element type	Data type	Parameter name (Unit)	Description
0x3F	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Settling Time (s)	Delay time for setting the on-target state. Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x47	3	Hexapod struts 1 to 6, axes A and B	INT	Default Direction for the Reference Move	Default direction for the reference move 0 = Automatic Detection Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x48	1	Hexapod struts 1 to 6, axes A and B	INT	Velocity-dependent offset	Velocity-dependent offset Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x49	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Velocity in closed-loop operation (Phys. Unit/s)	Velocity in closed-loop operation Limited by parameter 0xA Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x4A	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Maximum allowed acceleration	Maximum acceleration Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x4B	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Maximum allowed decelera-tion	Maximum delay Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x50	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Velocity for Reference Moves (Phys. Unit/s)	Maximum velocity for reference move Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x5A	0	Hexapod struts 1 to 6, axes A and B	INT	Numerator of the Servo-Loop Input Factor	Numerator and denominator of the servo-loop input factor The servo-loop input factor decouples the servo control parameters from the encoder resolution. The servo-loop input factor is independent of the factor for counts per physical length unit (0xE and 0xF). Numerator and denominator of the servo-loop input factor should not be changed. Is changed in the volatile memory for
0x5B	0	Hexapod struts 1 to 6, axes A and B	INT	Denumera-tor of the Servo-Loop Input Factor	

Parameter ID (hexa decimal)	Command level for write access	Affected element type	Data type	Parameter name (Unit)	Description
					axes A and B when a positioner type is assigned.
0x63		Hexapod struts 1 to 6, axes A and B	FLOAT	Distance between internal limit switch and hard stop (Phys. Unit)	Distance between the built-in limit switch and the hard stop Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x70	0	Hexapod struts 1 to 6, axes A and B	INT	Reference Signal Type	Reference signal type 0 = Direction-sensing reference switch. The signal level changes when passing the reference switch. 1 = Pulse signal with a pulse width of several nanoseconds (parameter 0x47 must be set correctly). 2 = Index pulse. The reference switch is approached via the negative limit of the travel range. 3 = Index pulse. The reference switch is approached via the positive limit of the travel range. 4 = No reference signal 5 = The reference signal is output at the negative limit switch. 6 = The reference signal is output at the positive limit switch. Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x71	0	Hexapod struts 1 to 6, axes A and B	INT	D-Term-Delay (No. of Servo Cycles)	D term delay 1 to 8 servo cycles Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x72	0	System	INT	Ignore Macro Error?	Ignore macro error? Determines whether the controller macro is stopped if an error occurs when it is running. 0 = Stop macro on error (default) 1 = Ignore macro error
0x77	0	Hexapod struts 1 to 6, axes A and B	INT	Use Limit Switches only for Reference Move	Should the limit switches only be used for reference moves? Is changed in the volatile memory for axes A and B when a positioner type is assigned.

Parameter ID (hexa decimal)	Command level for write access	Affected element type	Data type	Parameter name (Unit)	Description
0x78	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Distance between limit switch and the starting position (Phys. Unit)	Distance between the limit switch or hard stop and the starting position for the reference move to the index pulse Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x79	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Maximum distance for the motion to the index pulse (Phys. Unit)	Maximum distance for the reference move to the index pulse Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x94	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Notch Filter Frequency 1 (Hz)	Frequency of the first notch filter Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0x95	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Notch Filter Edge 1	Rise of the edge of the first notch filter Is changed in the volatile memory for axes A and B when a positioner type is assigned.
0xAC	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Lowpass Filter Frequency 1 (Hz)	Frequency of the first low-pass filter
0x03000100	2	Input signal channel	FLOAT	Polynomial 0 order of Range X (X = Index)	Coefficients of the polynomial for linearizing the analog inputs
0x03000200	2	Input signal channel	FLOAT	Polynomial 1st order of Range X (X = Index)	
0x03000300	2	Input signal channel	FLOAT	Polynomial 2nd order of Range X (X = Index)	
0x03000400	2	Input signal channel	FLOAT	Polynomial 3rd order of Range X (X = Index)	
0x03000500	2	Input signal channel	FLOAT	Polynomial 4th order of Range X (X = Index)	

Parameter ID (hexa decimal)	Command level for write access	Affected element type	Data type	Parameter name (Unit)	Description
0x03000600	2	Input signal channel	FLOAT	Polynomial 5th order of Range X (X = Index)	
0x03003320	1	Hexapod struts 1 to 6, axes A and B	INT	Type of Sensor	Signal type output by the position sensor 1 = Incremental sensor, AB quadrature signals (default setting) 3 = Incremental sensor, BiSS (32 bit) 4 = Absolute-measuring sensor, BiSS (32 bit) 7 = Incremental sensor, BiSS (24 bit) 8 = Absolute-measuring sensor, BiSS (24 bit)
0x03003330	1	Hexapod struts 1 to 6, axes A and B	FLOAT	Sensor Position Offset	Offset for correcting the signals of an absolute-measuring sensor
0x03003340	1	Hexapod struts 1 to 6, axes A and B	INT	Sensor Number of CRC Errors	Number of checksum errors when a sensor signal type with BiSS protocol is used
0x03003360	3	Hexapod struts 1 to 6, axes A and B	FLOAT	Sensor Home Position	Internal scaling/rounding of the sensor value
0x04000300	3	Input signal channel	FLOAT	ADC Range Minimum	Settings for the analog inputs of the C-887: ▪ Value range of the A/D converter ▪ Gain value ▪ Bit resolution ▪ Unit for display
0x04000400	3	Input signal channel	FLOAT	ADC Range Maximum	
0x04000700	3	Input signal channel	FLOAT	Hardware Gain	
0x04000B01	3	Input signal channel	INT	ADC Input Resolution	
0x04000E00	3	Input signal channel	CHAR	Display Unit	
0x07000000	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Range Limit Min	Limits of the permissible range for motion (in physical units)
0x07000001	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Range Limit Max	
0x07000601	0		CHAR	Axis Unit	Unit symbol

Parameter ID (hexa decimal)	Command level for write access	Affected element type	Data type	Parameter name (Unit)	Description
0x07030401	0	Hexapod axis	INT	Behaviour After Reference Move	Behavior of the hexapod's motion platform after the reference move For details, see "Motion of the Hexapod" (p. 31).
0x07030402	0	Hexapod axis	FLOAT	Target For Motion After Reference Move	
0x0D000200	2	Hexapod struts 1 to 6, axes A and B	INT	Hardware Name	
0x0D000700	2	System	CHAR	Device Name	Product name of the C-887
0x0D001000	1	System	CHAR	Customer Device Name	Freely selectable name affix for the C-887 Can be adapted by the user to distinguish between several C-887 in the same network or on the same PC. Is shown in the list of controllers found, e.g., when communication is established via TCP/IP.
0x0E000200	3	System	FLOAT	Servo Update Time	Servo cycle time in seconds
0x0E000900	0	System	INT	Pulse Length Factor	Multiplied by the cycle time (p. 348) Product corresponds to the pulse width of the impulse Default: 5
0x0E000B00	3	System	INT	Number of Input Channels	Number of analog input channels installed For details on the identifiers of the input channels, see "Commandable Elements" (p. 25). Corresponds to the response to TAC? (p. 254)
0x0E001600	0	System	INT	HID Device Button Mode	Behavior of the pushbuttons of the (C-887.MC2 or C-887.MC) manual control unit 0 = Pushbuttons trigger actions (stop, reference move; default) 1 = Pushbuttons do not trigger any actions For more information, refer to "Hexapod Motion" (p. 31).

Parameter ID (hexa decimal)	Command level for write access	Affected element type	Data type	Parameter name (Unit)	Description
0x13000004	3	System	INT	Max Wave Points	Total number of available points for waveforms Refer to "Wave Generator" (p. 99) for details.
0x1300010A	3	System	INT	Number Of Wave Tables	Number of wave tables for saving waveforms Refer to "Wave Generator" (p. 99) for details.
0x16000000	0	System	INT	Data Recorder Table Rate	Record table rate 1 to 10000 Default: 10 See also RTR (p. 240).
0x16000100	3	System	INT	Max. Number of Data Recorder Channels	Maximum number of data recorder tables
0x16000200	3	System	INT	Data Recorder Max. Points	Maximum number of all points of the data recorder tables
0x16000201	0	System	INT	Data Recorder Points Per Table	Number of points per data recorder table 1 to 10240 Default: 1024
0x16000300	3	System	INT	Channel Number	Number of data recorder tables
0x19000951	3	System	INT	Status	License status of the PIVeriMove hexapod software for collision checking (C-887.VM1, see "Optional Accessories" (p. 22)) -2 = license invalid -1 = configuration file invalid, e.g., formatted incorrectly 0 = not activated 1 = activated
0x19000952	0	System	FLOAT	Security Distance (mm)	Security distance for use in the PIVeriMove hexapod software for collision checking Parameter value must be ≥ 0
0x19001500	3	System	FLOAT	Maximum System Velocity (mm/s)	Maximum system velocity Refer to specifications in the user manual for the hexapod and "Motion of the Hexapod" (p. 31).

Parameter ID (hexa decimal)	Command level for write access	Affected element type	Data type	Parameter name (Unit)	Description
0x19001501	3	System	FLOAT	Minimum System Velocity (mm/s)	Minimum system velocity For details, see "Motion of the Hexapod" (p. 31).
0x19001502	3	System	FLOAT	Maximum System Accelera-tion (mm/s^2)	Maximum system acceleration For details, see "Motion of the Hexapod" (p. 31).
0x19001504	3	System	FLOAT	Path Control Step Size (mm)	Step size for calculating the dynamics profile of the platform motion For details, see "Motion of the Hexapod" (p. 31).
0x19001510	0	System	FLOAT	Trajectory Velocity (Phys. Unit/s)	Velocity, acceleration, and jerk for the motion platform of the hexapod For details, see "Profile Generator for Point-to-Point Motion" (p. 34).
0x19001511	0	System	FLOAT	Trajectory Accelera-tion (Phys. Unit/s)	
0x19001512	0	System	FLOAT	Trajectory Jerk (Phys. Unit/s)	
0x19001800	0	System	INT	Coordina-tion Mode	Coordination mode 0 = multi-axis mode 1 = Hexapod mode (default) The parameter value may not be changed.
0x19001900	0	System	INT	Trajectory Source	Source of the dynamics profile for MOV commands 0 = Dynamics profile is determined by profile generator (default) 1 = Dynamics profile is determined by consecutive `MOV` commands For details, see "Motion of the Hexapod" (p. 31).
0x19001901	0	System	INT	Trajectory Execution	Execution of the dynamics profile 0 = Dynamics profile is executed immediately (default) 1 = Dynamics profile is stored in a buffer before execution For details, see "Cyclic Transfer of Target Positions" (p. 36).

Parameter ID (hexa decimal)	Command level for write access	Affected element type	Data type	Parameter name (Unit)	Description
0x19001902	3	System	INT	Maximum Number of Trajectory Points	Maximum number of dynamics profile points For details, see "Cyclic Transfer of Target Positions" (p. 36).
0x19001903	0	System	INT	Threshold for Trajectory Execution	Threshold value for executing the dynamics profile For details, see "Cyclic Transfer of Target Positions" (p. 36).
0x19001904	3	System	INT	Current Number of Trajectory Points	Shows the current number of dynamics profile points in the buffer. For details, see "Cyclic Transfer of Target Positions" (p. 36).
0x19002000	0	System	INT	Configure Command Mode	Selection of the control source for the motion platform axes of the hexapod (X, Y, Z, U, V, W) 0 = GCS, see "Motion of the Hexapod" (p. 31) 1 = External: EtherCAT master. Only possible when the C-887 has an EtherCAT fieldbus interface.
0x19002001	0	System	INT	Start All Hexapod Wave Generators	Start behavior of the wave generators for the axes of the motion platform of the hexapod (X, Y, Z, U, V, W). Refer to "Wave Generator" (p. 99) for details.
0x19002010	3	System	FLOAT	Cycletime for Inter-polation in External Mode	Currently valid cycle time of the C-887 during control by an EtherCAT master.
0x19003000	1	Hexapod struts 1 to 6	FLOAT	Hexapod Position Vector A0_X	Position and direction vectors for the struts and joints of the hexapod
0x19003001	2	Hexapod struts 1 to 6	FLOAT	Hexapod Position Vector A0_Y	
0x19003002	1	Hexapod struts 1 to 6	FLOAT	Hexapod Position Vector A0_Z	
0x19003010	1	Hexapod struts 1 to 6	FLOAT	Hexapod Joint Vector A0_X	
0x19003011	1	Hexapod struts 1 to 6	FLOAT	Hexapod Joint Vector A0_Y	

Parameter ID (hexa decimal)	Command level for write access	Affected element type	Data type	Parameter name (Unit)	Description
0x19003012	1	Hexapod struts 1 to 6	FLOAT	Hexapod Joint Vector A0_Z	
0x19003100	1	Hexapod struts 1 to 6	FLOAT	Hexapod Position Vector B0_X	
0x19003101	1	Hexapod struts 1 to 6	FLOAT	Hexapod Position Vector B0_Y	
0x19003102	1	Hexapod struts 1 to 6	FLOAT	Hexapod Position Vector B0_Z	
0x19003110	1	Hexapod struts 1 to 6	FLOAT	Hexapod Joint Vector B0_X	
0x19003111	1	Hexapod struts 1 to 6	FLOAT	Hexapod Joint Vector B0_Y	
0x19003112	1	Hexapod struts 1 to 6	FLOAT	Hexapod Joint Vector B0_Z	
0x19003200	2	Hexapod struts 1 to 6	FLOAT	Hexapod Joint GLA	
0x19003201	2	Hexapod struts 1 to 6	FLOAT	Hexapod Joint GLB	
0x19003202	2	Hexapod struts 1 to 6	FLOAT	Hexapod Joint Type	
0x19003203	1	Hexapod struts 1 to 6	FLOAT	Hexapod Spindle Pitch	
0x19004000	1	System	INT	Check PowerGood Signal	Check of the Power Good signal 0 = Power Good signal is not checked 1 = Power Good signal is checked For details, see "Automatically Switching off the Servo Mode / Stopping the Motion" (p. 88).
0x19004001	1	System	INT	Axis Handling on Motion Error	Selection of the axes that are handled when a fault occurs 0 = Handling only for the affected axis 1 = Handling of all axes (default) For details, see "Automatically Switching off the Servo Mode / Stopping the Motion" (p. 88).

Parameter ID (hexa decimal)	Command level for write access	Affected element type	Data type	Parameter name (Unit)	Description
0x19004002	1	System	INT	Type of Axis Handling on Motion Error	Type of handling when fault occurs 0 = Stopping of the motion with STP (error code 10 is set), servo mode remains switched on 1 = Stopping the motion by switching off the servo mode (default) For details, see "Automatically Switching off the Servo Mode / Stopping the Motion" (p. 88).
0x19005000	1	System	INT	Stiffness Compensa-tion	Stiffness compensation. During motion commands, the stiffness compensation corrects the current position of the hexapod's motion platform. The correction is made depending on the defined load and stiffness, and is based on a theoretical model. To use the stiffness compensation, additional configuration files are required as well as an adjustment of the parameter values. If you want to use the stiffness compensation, contact our customer service department (p. 343).
0x19005001	1	System	FLOAT	Mass of Load	
0x19005002	1	System	FLOAT	Center of Gravity	
0x19005003	1	System	FLOAT	Tensor Matrix	
0x19005004	1	System	FLOAT	Direction to GeoCenter	
0x19006000	0	System	FLOAT	Reduction Factor for TRA? Response	Factor for limiting the response to TRA? 0 to 1 The parameter **does not** influence the actual soft limits, but limits only the position values displayed by TRA? (p. 257). For details, see "Travel Range and Soft Limits" (p. 31).
0x19006001	2	System	INT	Enable Gradient Search	Availability of fast alignment routines for gradient search (FDG command) 0 = gradient search not available 1 = gradient search available See also option R-FMP-GSM in "Optional Accessories" (p. 22).
0x19006002	0	System	INT	Check Calculated Trajectory Before Motion	Activation state of the dynamics profile check 0 = check deactivated 1 = check activated For details, see "Profile Generator for Point-to-Point Motion" (p. 34) and "Troubleshooting" (p. 335).

Parameter ID (hexa decimal)	Command level for write access	Affected element type	Data type	Parameter name (Unit)	Description
0x1A002000	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Max Enc Vel	Safety shutdown in the event of a fault For details, see "Configuring Safety Shutdown" (p. 95).
0x1A002100	0	Hexapod struts 1 to 6, axes A and B	INT	Activate Motor Stuck Check	
0x1A002200	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Minimal MotorOut to Move Axis	
0x1A002300	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Velocity Threshold under which Axis is consi-dered not moving	
0x1A002400	0	Hexapod struts 1 to 6, axes A and B	FLOAT	Time Period for which Axis is not yet consi-dered not moving	
0x20001C00	1	System	INT	Fast Alignment Axis Signal Type	Type of position signal for fast alignment routines Fast alignment routines make calculations based on the position of the axes involved. Possible types: 0 = current position, based on encoder signals 1 = dynamics profile created by the profile generator of the C-887 (default) The use of the current position can be useful, for example, if the signal maximum found is to be approached at the end of an area scan routine. In this case, the deviation when approaching the signal maximum can be minimally reduced compared to the deviation when using the dynamics profile. For details, refer to E712T0016 "Fast Multi-Channel Photonics Alignment (FMPA)".

10 Maintenance

In this Chapter

10.1 Cleaning the C-887

NOTICE

Short circuits or flashovers!

The C-887 contains electrostatic-sensitive devices that can be damaged by short-circuiting or flashovers when cleaning fluids penetrate the housing.

- Before cleaning, disconnect the C-887 from the power source by removing the mains plug.
- Prevent cleaning fluid from penetrating the housing.

- When necessary, clean the surfaces of the C-887's housing using a cloth dampened with a mild cleanser or disinfectant.

10.2 Updating Firmware and Configuration Files

10.2.1 General Notes on Updating Firmware and Configuration Files

NOTICE

Damage from unintentional position changes!

The limit value for the load of the hexapod determined by the simulation program only applies when servo mode is switched on (p. 60) for the axes of the motion platform. The maximum holding force is based on self-locking of the actuators in the hexapod struts when the servo mode is switched off and is lower than the limit value when the servo mode is switched on (see manual for the hexapod).

When the actual load of the hexapod exceeds the maximum holding force based on the self-locking of the actuators, unintentional position changes of the hexapod can occur in the following cases:

- Switching off the C-887
- Rebooting the C-887
- Switching the servo mode off for the axes of the motion platform of the hexapod, e. g., by using the **E-Stop** socket (p. 91)

As a result, collisions are possible between the hexapod, the load to be moved, and the surroundings. Collisions can damage the hexapod, the load to be moved or the surroundings.

➢ Make sure that the actual load of the motion platform of the hexapod does not exceed the maximum holding force based on the self-locking of the actuators before you switch off the servo mode, reboot or switch off the C-887.

NOTICE

Malfunction after switching off during a firmware update!

If you switch off the C-887 during a firmware update, the firmware and configuration files will be transferred to the C-887 incompletely or incorrectly. Incomplete or incorrect firmware and configuration can prevent the C-887 from booting.

- Do **not** switch off the C-887 while the firmware is being updated.
- Note the IP address of the C-887 when establishing the connection with the PIFirmwareManager.
- If the C-887 has been switched off while the firmware was being updated and still boots afterwards, repeat the update of the firmware.
- If the C-887 has been switched off while the firmware was being updated and no longer boots afterwards:
  - Do **not** switch off the C-887.
  - Contact our customer service department (p. 343).

INFORMATION

The firmware of the C-887 consists of several components that can be updated separately with the PIFirmwareManager program. Combining different versions of individual components must be approved by PI.

- Before you update the firmware of the C-887, check the versions of the firmware components by querying with VER? (p. 261).
- For information on approved version combinations, contact our customer service department (p. 343).
- Update the individual firmware components in separate steps. When our customer service department has specified a particular sequence for updating the components, observe this sequence.

INFORMATION

To update the firmware, the C-887 and PC must communicate via the TCP/IP interface.

10.2.2 Getting Current Firmware and Configuration Files

Getting current firmware and configuration files of the C-887

1. Have the following information ready:
   - Read the device identification string of the C-887 with the `*IDN?` command.
   - Read the versions of the firmware components with the `VER?` command.
   - Read out the hexapod type that the controller is adapted to with the `CST?` command.
2. Contact our customer service department (p. 343) to obtain current versions of the firmware components and configuration files as well as information on approved version combinations.

10.2.3 Updating Firmware

Tools and accessories

- PC with Windows operating system that has been prepared as follows:
  - The PIFirmwareManager program is installed. For further information, see "Installing the PC Software" (p. 56).
  - The current firmware and/or configuration files that you have obtained from our customer service department (p. 319) are in a directory on the PC.

Requirements

- ✓ You have read and understood the general notes on updating firmware and configuration files (p. 318).
- ✓ You have made all necessary preparations for communication via the TCP/IP interface, see "Establishing Communication via TCP/IP in the PC-Software" (p. 76).
- ✓ The C-887 is switched on and the starting procedure of the C-887 has finished (p. 71).
- ✓ The PC is switched on.

Updating the firmware of the C-887

1. Start the PIFirmwareManager program on the PC via the ***All Programs > PI > PIFirmwareManager*** start menu item.

   The ***PIFirmwareManager*** window opens.

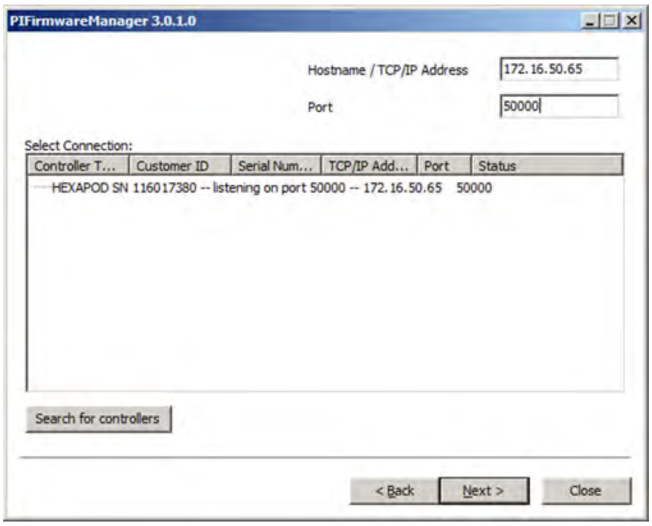
   – Click the ***Next >*** button.
2. Establish communication between the C-887 and the PC.

   a) Click the ***Search for controllers*** button.

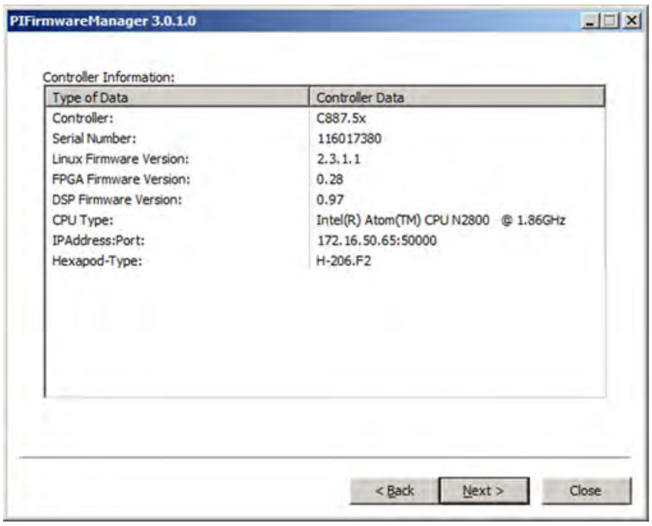
   b) If your C-887 is shown in the ***Select Connection:*** field, mark the corresponding line in the list.

   c) If your C-887 is not displayed in the ***Select Connection:*** field but its current IP address is known, enter the address in the ***Hostname / TCP/IP Address*** field and *50000* in the ***Port*** field.

d) Click the ***Next >*** button.

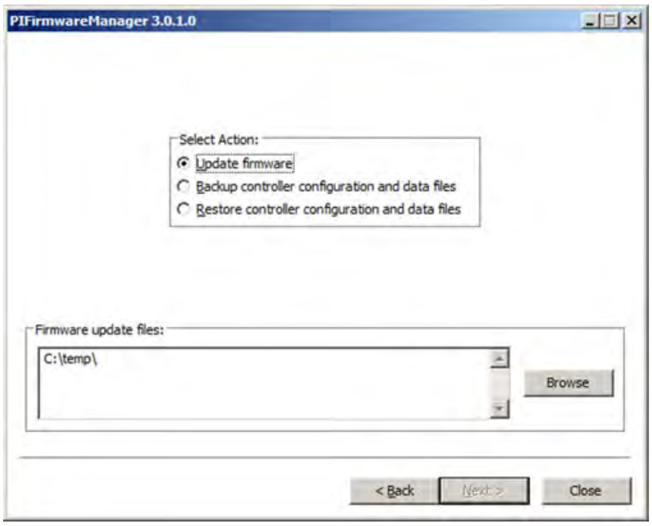


3. Check whether the displayed information matches the information that you have received with the new firmware from our customer service department.

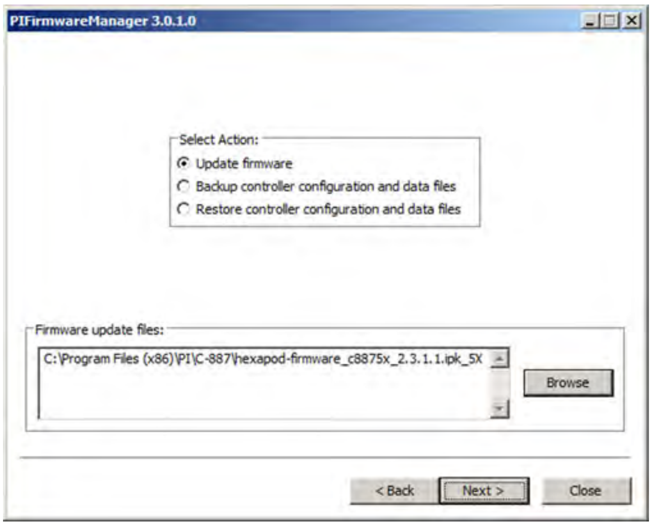


– Click the ***Next >*** button.

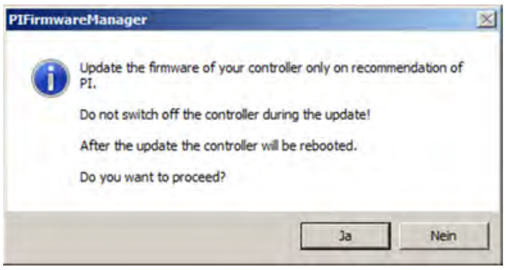
4. Click the ***Update firmware*** option in the ***Select Action:*** field.



5. Select the file for the firmware component to be updated.
   a) Click the ***Browse*** button. A file selection window opens.
   b) In the file selection window, go to the directory that has the files that you have received from the customer service department.
   c) In the file selection window, select the file for the firmware component to be updated, e.g., hexapod-firmware_c8875x_x.x.x.x.ipk_5X.
   d) Click the ***Open*** button in the file selection window to confirm the selection. The file is shown in the ***Firmware update files:*** field in the PIFirmwareManager.
6. Start the transfer of the firmware to the C-887.



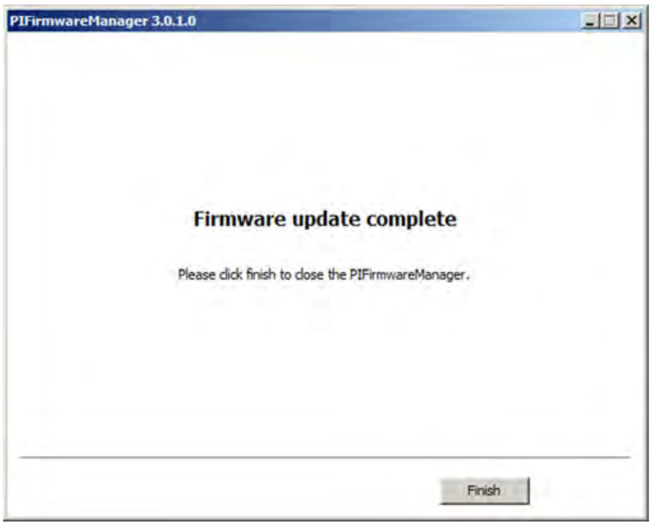
a) Click the ***Next >*** button. A dialog window with information opens.



b) Click ***Yes*** in the dialog window to start transferring the firmware to the C-887.

The update progress is displayed. The update finishes when the C-887 reboots.

7. Click the ***Finish*** button.



The PIFirmwareManager closes.

8. If you want to update another firmware component of the C-887, repeat step 1 to 7.

10.2.4 Updating Configuration Files

NOTICE

Malfunction after incorrect changing of configuration files!
If you transfer an incorrect or incomplete tar archive to the C-887, you can render the C-887 inoperable.

- Note the IP address of the C-887 when establishing the connection with the PIFirmwareManager.
- When you edit the contents of the tar archive on the PC, do not change the directory structure or the names of directories and existing files.
- Change and supplement configuration files in the tar archive only in consultation with PI.
- Do not switch off the C-887 while the tar archive is being retransferred.
- If incorrect configuration files have been transferred to the C-887:
  - Do **not** switch off the C-887.
  - Contact our customer service department (p. 343).

You can update various configuration files of the C-887 and transfer additional configuration files to the C-887.

Direct access to the configuration files of the C-887 is not possible. You first have to copy the configuration file of the C-887 as a tar archive to the PC. You update and supplement the configuration files in the tar archive. You then transfer the modified tar archive from the PC to the C-887.

INFORMATION

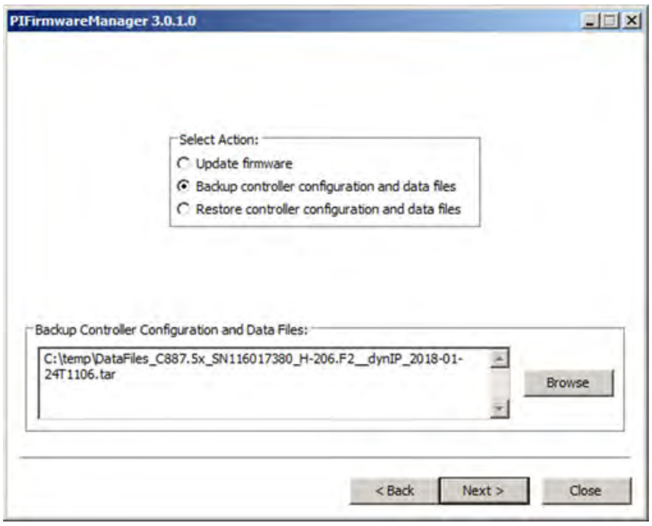
Application tip:
You can "duplicate" a C-887 by copying its configuration files to the PC as a tar archive and then transferring the archive from the PC to another C-887 without changing. The serial number of the C-887 to which the copied data is transferred remains unchanged.

Updating configuration files of the C-887

1. To copy the tar archive with the configuration files of the C-887 to the PC, start the PIFirmwareManager. For this purpose, do steps 1 to 3 from the "Updating Firmware" (p. 319) instructions.
2. In the ***Select Action:*** field, click the ***Backup controller configuration and data files*** option.

   In the ***Backup Controller Configuration and Data Files*** field, a suggestion is given for the directory path on the PC and the name of the tar archive.

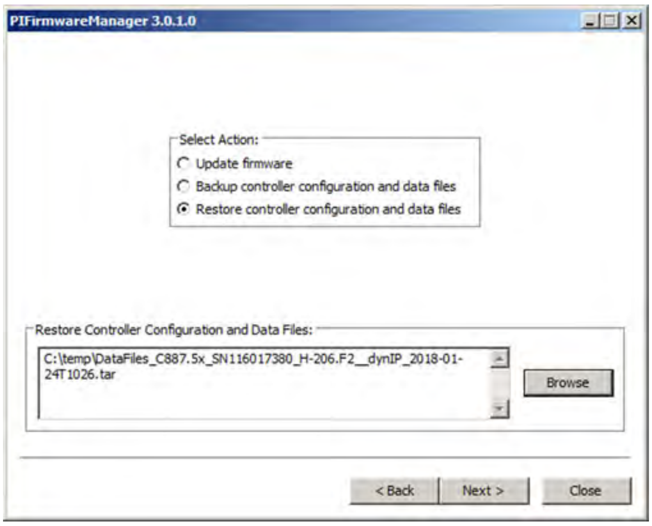
- Check whether the suggested directory exists on the PC.
- If you want to change the directory on the PC and/or the name of the tar archive: Click the ***Browse*** button and select anew directory and/or a new name.
- Do **not** change the file extension .tar.



3. Start the transfer of the tar archive to the PC by clicking the ***Next >*** button.

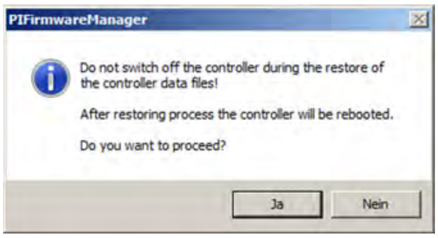
   The progress of the transfer is displayed. The transfer to the PC has ended when *Backup of controller data files complete* is displayed in the PIFirmwareManager.
4. Click ***Finish*** to close the PIFirmwareManager.
5. Update and supplement the configuration files in the tar archive on the PC.
   a) On the PC, open the directory in which the PIFirmwareManager has stored the tar archive with the copies of the configuration files.
   b) Update and supplement the configuration files in the tar archive according to the instructions that you received with the new configuration files from our customer service department.
6. To retransfer the modified tar archive from the PC to the C-887, start the PIFirmwareManager again. For this purpose, do steps 1 to 3 from the "Updating Firmware" (p. 319) instructions.
7. In the ***Select Action:*** field, click the ***Restore controller configuration and data files*** option.
8. Select the tar archive on the PC to be transferred.

– Click the ***Browse*** button.



9. Start retransferring the tar archive to the C-887.

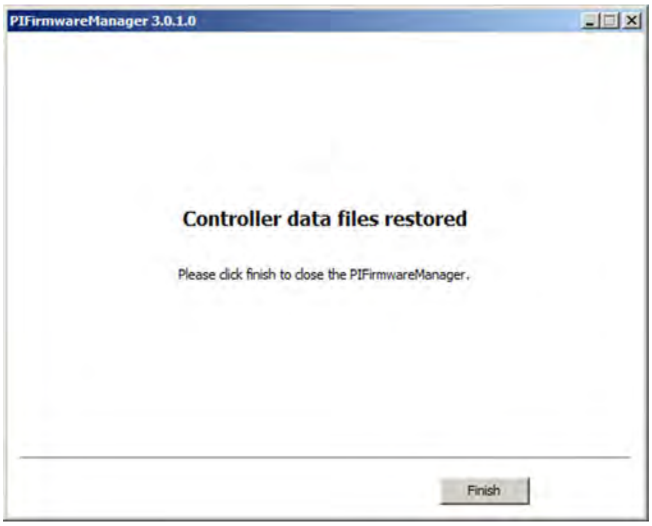
   a) Click the ***Next >*** button. A dialog window with information opens.



   b) Click ***Yes*** in the dialog window to start transferring the tar archive to the C-887.

   The retransfer progress is displayed. Retransfer finishes after the C-887 reboots.

10. Click the ***Finish*** button.



The PIFirmwareManager closes.

10.3 Maintaining and Checking the Hexapod

10.3.1 Doing a Maintenance Run

CAUTION

Risk of crushing by moving parts!

Risk of minor injuries from crushing between the moving parts of the hexapod and a stationary part or obstacle.

- Keep your fingers away from areas where they could be caught by moving parts.

NOTICE

Damage due to collisions!

Collisions can damage the hexapod, the load to be moved, and the surroundings.

- Make sure that no collisions are possible between the hexapod, the load to be moved, and the surroundings in the workspace of the hexapod.
- Do not place any objects in areas where they can be caught by moving parts.
- Stop the motion immediately if a controller malfunction occurs.

Frequent motion over a limited travel range can cause the lubricant to be distributed unevenly on the drive screws of the hexapod struts. This can result in increased wear. A maintenance run across the entire travel range of the hexapod struts redistributes the lubricant evenly.

INFORMATION

The more motion is done over a limited travel range, the shorter the time must be between the maintenance runs.

- Carry out the maintenance run at appropriate intervals according to the requirements of your application.

Requirements

- ✓ You have read and understood the general notes on startup (p. 67).
- ✓ You have installed the hexapod system correctly (p. 55).
- ✓ You have read and understood the user manual for the hexapod.

✓ If you want to do the maintenance run with PIMikroMove:
  - PIMikroMove is installed on the PC (p. 56).
  - You have read and understood the PIMikroMove manual. The links to the software manuals are in the A000T0081 file on the PI software CD.
  - You have established communication between the C-887 and the PC with PIMikroMove via the TCP/IP interface (p. 76) or the RS-232 interface (p. 79).
  - You have read and understood the section "Starting Motion" (p. 81).

Doing a Maintenance Run

1. Set the velocity for the motion platform of the hexapod with the `VLS` command (p. 262) to about 30 % of the default value which is valid after switching on or rebooting the C-887.
2. Command hexapod motion to the following target position:

   Z = Greatest commandable position (query with the `TMX?` command (p. 256) possible)

   X = Y = U = V = W = 0
3. Command hexapod motion to the following target position:

   Z = Smallest commandable position (query with the `TMN?` command (p. 255) possible)

   X = Y = U = V = W = 0
4. Repeat steps 2 and 3 three times each.
5. Command hexapod motion to the target position X = Y = Z = U = V = W = 0.

10.3.2 Doing a Strut Test

CAUTION

Risk of crushing by moving parts!
Risk of minor injuries from crushing between the moving parts of the hexapod and a stationary part or obstacle.

➢ Keep your fingers away from areas where they could be caught by moving parts.

NOTICE

Damage due to collisions during the strut test!

The hexapod moves unpredictably during a strut test. A collision check or prevention does **not** take place, even if a configuration for preventing collisions was stored on the C-887 with the PIVeriMove hexapod software for collision checking. Soft limits that have been set for the motion platform of the hexapod with the `NLM` (p. 234) and `PLM` (p. 236) commands are ignored during a strut test.

As a result, collisions are possible between the hexapod, the load to be moved, and the surroundings. Collisions can damage the hexapod, the load to be moved, and the surroundings.

- Make sure that collisions are not possible between the hexapod, the load to be moved, and the surroundings during a strut test of the hexapod.
- Do not place any objects in areas where they could be caught by moving parts during a strut test.
- Pay attention to the hexapod during a strut test in order to intervene quickly in the case of malfunctions.

NOTICE

Damage from unintentional position changes during the strut test!

The hexapod strut can reach a limit switch during a strut test. As a result, servo mode is automatically switched off for the axes of the motion platform of the hexapod.

When the actual load of the hexapod exceeds the maximum holding force based on the self-locking of the actuators, switching off the servo mode for the axes of the motion platform of the hexapod can cause unintentional position changes of the hexapod.

As a result, collisions are possible between the hexapod, the load to be moved, and the surroundings. Collisions can damage the hexapod, the load to be moved, and the surroundings.

- Make sure that the actual load of the motion platform of the hexapod does not exceed the maximum holding force based on the self-locking of the actuators before you start a strut test.

INFORMATION

Recommendations for performing a strut test:

- If possible command the hexapod to move to the reference position before a strut test, in order for each hexapod strut to have the greatest possible travel range in the positive and negative direction of motion available.
- Set the appropriate velocity with the `VLS` command (p. 262) before a strut test.

If the hexapod system malfunctions, e.g., motion error (error code 1024), the following tests on the hexapod struts can simplify error diagnosis:

- Measurement of the impulse response of the hexapod strut. Signs of malfunctions are:

  - Significant overshooting of the current position
  - Oscillations of the current position
  - Significant deviation between the graphs of the current position and the target position

- Measuring the motor control during one drive screw revolution of the hexapod strut (dimensionless value of -32767 to 32768; the sign corresponds to the direction of movement); among other things, enables assessment of the control reserve. Signs of malfunctions are:
  - Motor control is continuously at >70 % of the maximum value
  - Unusually high motor control value at a particular point in the travel range

The following tools can be used for the tests:

- ***Hexapod Service Tools*** window in PIMikroMove

INFORMATION

As an alternative to the impulse response, you can also record and evaluate a step response of the hexapod strut. After the step has been executed, the hexapod strut does **not** move back to the initial position, however. If a step is executed several times, the hexapod strut can reach the limit switch. When the limit switch is reached, servo mode is switched off for the axes of the motion platform of the hexapod.

Requirements

✓ You have read and understood the general notes on startup (p. 67).

✓ You have installed the hexapod system correctly (p. 55).

✓ You have read and understood the user manual for the hexapod.

✓ PIMikroMove is installed on the PC (p. 56).

✓ You have read and understood the PIMikroMove manual. The links to the software manuals are in the A000T0081 file on the PI software CD.

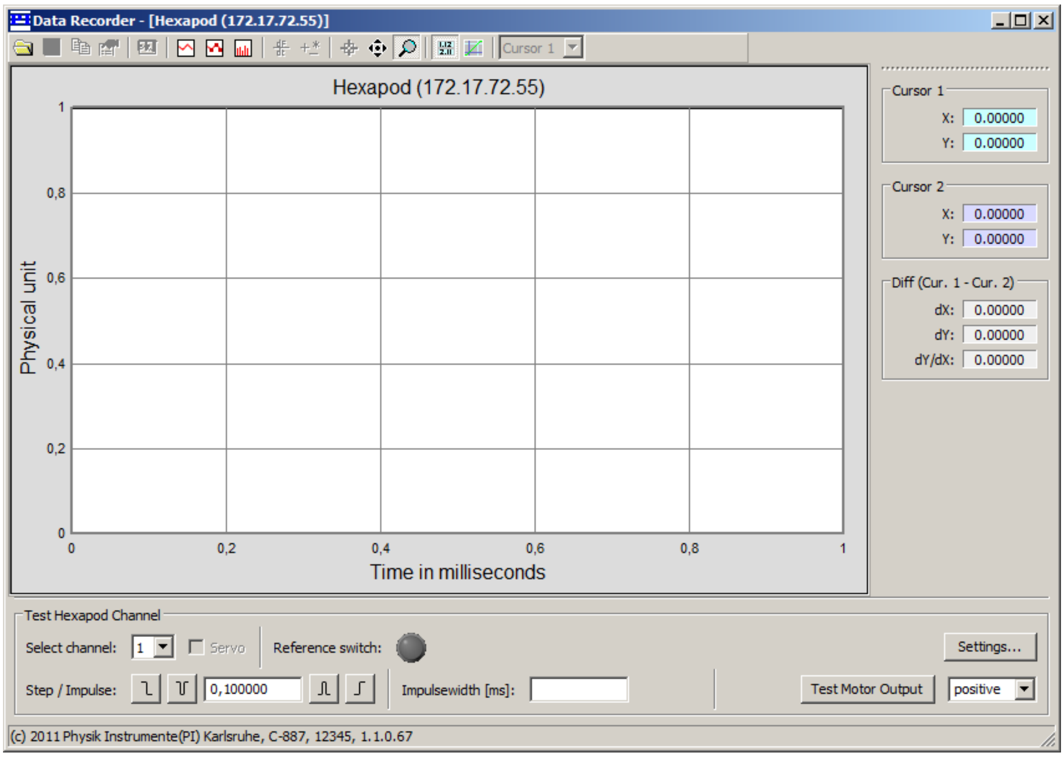
✓ You have established communication between the C-887 and the PC with PIMikroMove via the TCP/IP interface (p. 76) or the RS-232 interface (p. 79).

Doing the strut test in the *hexapod Service Tools* window of PIMikroMove

1. In the PIMikroMove main window, select the ***C-887 > Show service tools...*** menu item .

   A window opens with information on possible damage to equipment when the ***Hexapod Service Tools*** is used.
2. In the information window, click ***Show service tools***.

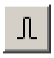
The ***Hexapod Service Tools*** window opens.



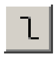
3. Record an impulse response for a hexapod strut in the ***Hexapod Service Tools*** window:

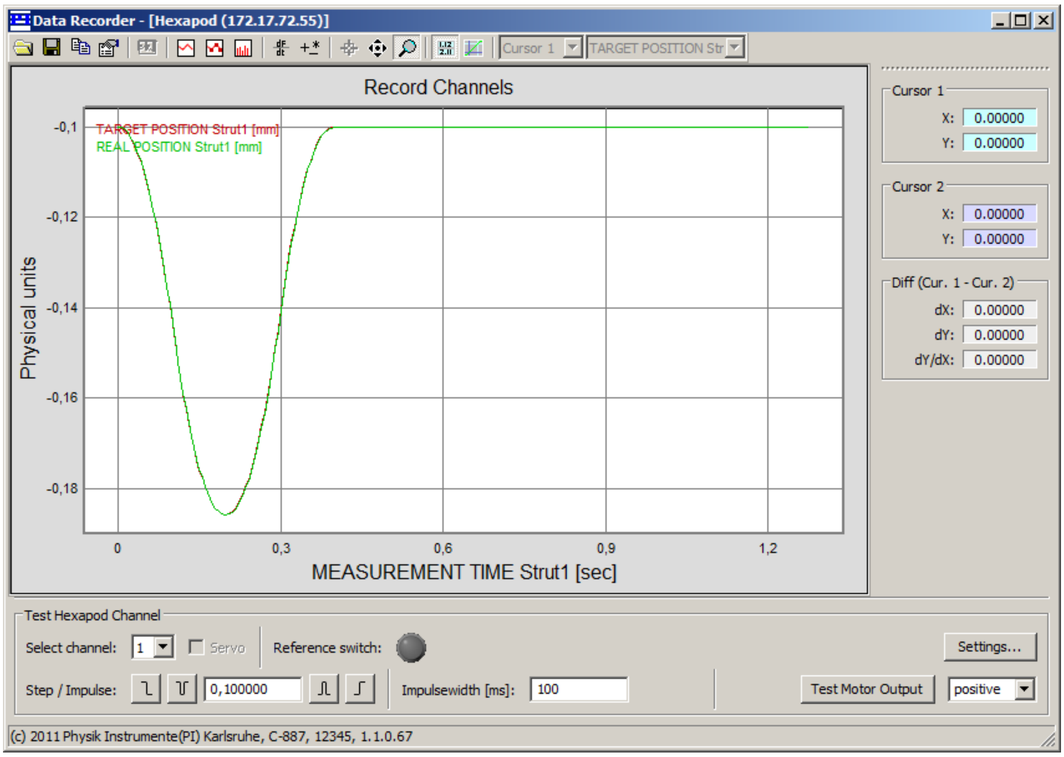
   a) In the ***Select channel*** field, select the strut for which the impulse response is to be recorded.

   b) In the ***Impulsewidth [ms]*** field, enter the pulse width of the impulse in milliseconds.

   c) Click the  or  button to start the impulse in a positive or negative direction. Starting the impulse also starts recording of the current position and the target position of the hexapod strut. The hexapod strut moves according to the set pulse width of the impulse and then returns to the initial position.

   d) Evaluate the impulse response using to the curves in the graphic field of the ***Hexapod Service Tools*** window (see following figure).

A step response with a corresponding recording can be started with the  or  button. The step size of the step is determined by the entry in the field between the buttons for starting the impulses.



4. Record the motor control during one drive screw revolution of a hexapod strut in the ***Hexapod Service Tools*** window:

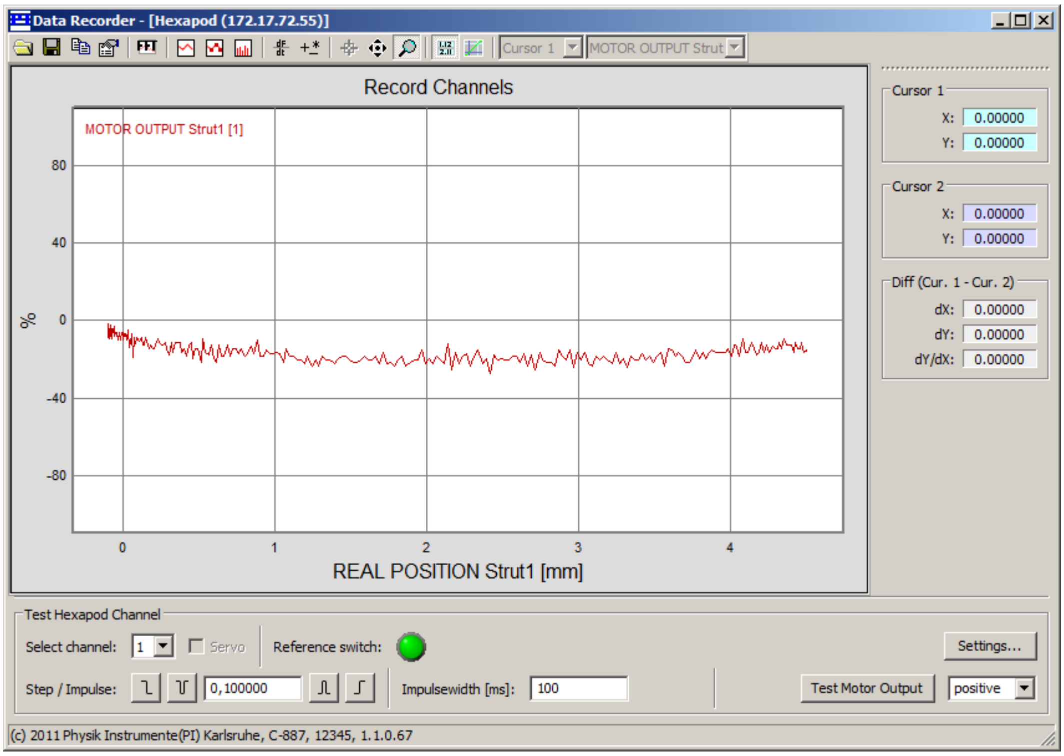
   a) In the ***Select channel*** field, select the strut for which the motor control is to be recorded.

   b) Select the direction of rotation (positive or negative) for the drive screw in the field on the right-hand side of the ***Test Motor Output*** button.

   c) Click the ***Test Motor Output*** button to start one drive screw revolution of the hexapod strut in the selected direction.
   Starting the drive screw revolution also starts the recording of the current motor control and the current position of the hexapod strut. The hexapod strut does **not** return to the initial position after the drive screw revolution.

   d) Evaluate the recorded motor control according to the curve in the graphic field of the ***Hexapod Service Tools*** window (see following figure).

If a drive screw revolution is executed several times in the same direction, the hexapod strut can reach the limit switch. When the limit switch is reached, servo mode is switched off for the axes of the motion platform of the hexapod.



5. If the test results indicate a malfunction:

   a) Do **not** operate the hexapod system further.

   b) Communicate the test results for the error diagnosis to our customer service department (p. 343).

11 Troubleshooting

Problem	Possible causes	Solution
C-887 does not boot	Power supply not connected correctly	➢ Check whether the C-887 is correctly connected to the power supply (p. 59).
	Firmware and/or configuration files incomplete or incorrect	1. Do not switch the C-887 off. 2. Try to determine the last valid IP address of the C-887. 3. Contact our customer service department (p. 343).

Problem	Possible causes	Solution
The hexapod does not move.	The cable is defective or is not connected correctly	➢ Check the cable connections.
The hexapod does not move.	Power supply of the hexapod is interrupted	➢ Check the power supply cable. ➢ If applicable, check the power adapter of the hexapod. ➢ Check the Power Good signal of the hexapod. Options: – Send the `DIA?` command (p. 154). – In PIMikroMove, open the ***Diagnostic Information*** window by choosing ***C-887 > Show diagnostic information...*** on the menu
The hexapod does not move.	Only C-887.522, .523, .532, .533 models: Connection of the **E-Stop** socket prevents motion from being triggered	➢ Check the Power Good signal and the activation state of the 24-V output for the hexapod (**24 V Out 7 A**). Options: – Send the `DIA?` command (p. 154). – In PIMikroMove, open the ***Diagnostic Information*** window by choosing ***C-887 > Show diagnostic information...*** on the menu ➢ Connect the **E-Stop** socket with external hardware according to the requirements of your application; for details see "Using the E-Stop Socket" (p. 91).
The hexapod does not move.	The hexapod does not have a Power Good signal, but the Power Good signal check is activated.	1. Send the `ERR?` command and check the error code that is returned. 2. If the response to `ERR?` contains error code 500 or 66: a) Check whether your hexapod is equipped with a Power Good signal (pin 59 of the **Hexapod**

Problem	Possible causes	Solution
		socket (p. 356)). The signal is normally missing in older models (e.g., H-840.D1), which are connected via an adapter from MDR68 to HD D-sub 78. You can find this information, for example, in the "Pin Assignment" chapter in the user manual of the hexapod. You can also use the `DIA?` command (p. 154). b) Query the activation state of the Power Good signal check by sending the following command: `SPA? 1 0x19004000` Response is 0: Power Good signal is not checked Response is 1: Power Good signal is checked 3. If your hexapod is not equipped with a Power Good signal, but the Power Good signal check is activated in the C-887, send the following command to deactivate the check: `SPA 1 0x19004000 0` For checking the Power Good signal, see also "Automatically Switching off the Servo Mode / Stopping the Motion" (p. 88).
The hexapod does not move.	Incorrect configuration of the C-887: ID chip of the hexapod not read out	➢ Connect the hexapod only when the controller is switched off. ➢ When the firmware has finished booting, send the `CST?` command (p. 154) to check whether the installed configuration has to be activated by rebooting the controller. A reboot is necessary when the response is "NOSTAGE" for the X, Y, Z, U, V, and W axes. The controller can be rebooted with the `RBT` command (p. 239). ➢ Send the `ERR?` command (p. 164). If the response to `ERR?` contains the error code 233 or 211, the configuration for the hexapod is missing in the controller. Contact our customer service department (p. 343) in order to receive a suitable configuration file. For the installation of the new configuration file, see "Updating Firmware and Configuration Files" (p. 318). ➢ Send the `VER?` command (p. 261) to check the information for the hexapod type, serial number, and manufacturing date saved on the ID chip. Example for the response: `IDChip: H-811.F-2 SN123456789 20/1/2016`

Problem	Possible causes	Solution
The hexapod does not move.	Incorrect configuration of the C-887: The hexapod does not have an ID chip	1. When the firmware has finished booting, send the CST? command (p. 154) to read out the hexapod type to that the controller is adapted to. 2. If the response to CST? does not match the connected hexapod, manually load the suitable configuration by sending the following command: `DBG? choosehexapod` *type* where *type* is the exact product name of the connected hexapod. The controller reports "1" (loading of configuration was successful) or "0" (loading was not successful). Depending on the response of the controller, further steps are necessary. If the controller has reported "1": 1. Send the RBT command (p. 239) to activate the new configuration by rebooting the controller. 2. When the firmware has finished booting, send the CST? command to check whether the controller is now adapted to the correct hexapod type. 3. If the response to CST? shows the correct hexapod type, you can proceed with starting your system. Otherwise, contact our customer service department (p. 343). If the controller has reported "0": 1. Send the ERR? command (p. 164). If the response to ERR? contains the error code 233 or 211, the configuration for the hexapod is missing in the controller. 2. Contact our customer service department (p. 343) in order to receive a suitable configuration file. For the installation of the new configuration file, see "Updating Firmware and Configuration Files" (p. 318). 3. Repeat the loading of the configuration (see above).
The hexapod does not move.	Incorrect command or incorrect syntax	➢ Send the ERR? command (p. 164) and check the error code that is returned.
The hexapod does not move.	Commanded position is outside the current travel range limits	Due to rounding, the response to TRA? may indicate a position that cannot be reached. Commanding such a position fails and generates error code 7 ("Position out of limits"). Therefore, you can limit the response to TRA? with a factor so that only positions that can actually be commanded are displayed:

Problem	Possible causes	Solution
		➢ Use SPA to set the ***Reduction Factor for TRA? Response*** parameter (0x19006000) to a suitable value between 0 and 1.
The hexapod does not move.	The motion platform is located at a position outside the travel range limits. Commanding a permitted target position generates error code 7 ("Position out of limits").	1. Send the `ERR?` command (p. 164) and check the error code that is returned. 2. Send the `POS?` command to check the current position of the motion platform. If error code 7 is reported and the current position is outside the travel range limits for at least one axis, the following steps are necessary depending on the sensor type of the hexapod (incremental or absolute measuring). **If the hexapod is equipped with incremental sensor:** 1. Check your system and make sure that all axes can be moved safely. 2. Start a reference move for the hexapod. **If the hexapod is equipped with absolute-measuring sensors:** 1. Check your system and make sure that all axes can be moved safely. 2. Use the parameter ***Check Calculated Trajectory Before Motion*** to temporarily deactivate the automatic check of the dynamics profile (p. 34). **Note:** By deactivating the automatic check of the dynamics profile, you are temporarily deactivating a safety function of the C-887. If the automatic check is deactivated, you are responsible for commanding the motion platform in the right direction and avoiding collisions. Send `SPA 1 0x19006002 0` to deactivate the automatic check. 3. Switch servo mode on for all axes with the `SVO` command (p. 252). 4. Command the motion platform to a position that lies in the permissible workspace and is approx. 1 to 5 µm away from the travel range limits. For this purpose, you can use one of the following commands: `MOV`, `MVR`, `MRW`, `MRT`. After successfully approaching the commanded position, the dynamics profile check is automatically reactivated. Notes: If approaching the commanded position fails, the dynamics profile check remains **deactivated.** The dynamics profile check is also reactivated with

Problem	Possible causes	Solution
		the `HLT`, `FRF`, `STE`, `IMP`, `VMO?` commands. 5. Check your system again: a) Send the `ERR?` command and check the error code that is returned. b) Send the `POS?` command to check the current position of the motion platform. c) Send the `SPA? 1 0x19006002` command to check the activation state of the dynamics profile check. If error code 0 is reported, the current position is within the travel range limits, and the dynamics profile check is activated, your system is ready for operation again. If one of these requirements is not fulfilled: 1. Repeat the steps mentioned above. 2. If the procedure remains permanently unsuccessful, contact our customer service department (p. 343).
The hexapod does not move.	Servo mode was switched off automatically	Axis motion is only possible when servo mode is switched on. Switching servo mode off stops the motion. 1. Send the `SVO?` (p. 253) command to check the activation state of the servo mode. 2. Send the `ERR?` command (p. 164) and check the error code that is returned. For details on possible error codes and their causes, see "Protective Functions of the C-887" (p. 88).
The hexapod does not move.	Wave generator output stops abruptly	During the wave generator output, the C-887 continuously checks whether the motion is still possible. In the following cases, the C-887 stops the motion abruptly and sets an error code: ▪ The target positions to be output cannot be reached. ▪ The necessary velocity cannot be achieved. ▪ The motion would cause a collision. ➢ Define suitable waveforms; for details, see "Creating a Waveform in the Wave Table" (p. 103).

Problem	Possible causes	Solution
The hexapod does not move.	▪ Foreign body has entered the drive screw ▪ Faulty motor ▪ Sensor defective ▪ Blocked or broken joint ▪ Load too heavy	The reference move is not executed successfully or servo mode is switched off automatically during motion (see "Switching Servo Mode off Automatically / Stopping Motion" (p. 88)). If possible: ➢ Carry out a strut test (p. 328) and communicate the test results to our customer service department (p. 343). ➢ Execute either motion in the Z axis or the motion at which the error occurs and record the current position (RecOption 2), the target position (RecOption 1), and the motor control value (RecOption 73) for the hexapod struts with the data recorder. Communicate the test results to our customer service department (p. 343).
Reduced repeatability of the positioning of the hexapod	▪ Warped base plate ▪ Warped motion platform	➢ Mount the hexapod on a flat surface (refer to the user manual for the hexapod). ➢ Only fix loads with a flat surface to the motion platform.
Reduced repeatability of the positioning of the hexapod	Poor lubrication because of small movements over a long period of time	➢ Perform a maintenance run at regular intervals (p. 327).
Reduced repeatability of the positioning of the hexapod	External disturbances	➢ Make sure that no vibrations are transmitted to the system. ➢ Make sure that the motion of the motion platform is not affected by forces resulting, for example, from cables being dragged along. ➢ Make sure that the system is in a thermal equilibrium.
Poor travel accuracy	▪ Unsuitable parameter settings for control ▪ Velocity, acce-leration too high or too low ▪ Unsuitable dynamics profile	➢ Adhere to the specifications of the hexapod when specifying the target position, velocity, and acceleration. It often helps to combine a lowest possible velocity with a highest possible acceleration. ➢ Set the parameters for the servo control (e.g. P, I, D term) appropriately. Typically, optimization is determined empirically, i.e., you observe the behavior of the hexapod with different values. ➢ If you specify the dynamics profile by successive MOV commands or via the EtherCAT interface,

Problem	Possible causes	Solution
	▪ Changed system behavior due to an increasing ease of movement	observe the information in "Cyclic Transfer of Target Positions" (p. 36). ➢ If you use the wave generator: Define suitable waveforms; for details, see "Creating a Waveform in the Wave Table" (p. 103).

Problem	Possible causes	Solution
Communication with the C-887 failed	Communication cable is incorrect or defective	➢ Check the cable. – Use a null modem cable for the RS-232 connection. – Use the straight-through network cable for TCP/IP connection via a hub or a router (with DHCP server). – Use the crossover network cable for direct connection with the Ethernet connection socket of the PC. ➢ If necessary, check whether the cable works on a fault-free system.
	Communication interface is not correctly configured	**When using the RS-232 interface:** ➢ Check the port settings, the baud rate, and the handshake setting of the PC. **When using the TCP/IP connection:** ➢ Connect the controller to the network **before** you switch it on. Otherwise, you will have to switch the controller off and on again. ➢ Check the network settings (p. 72). ➢ Make sure that the network is not blocked for unknown devices. ➢ Make sure that the network traffic to the C-887 is not blocked by a firewall. ➢ Make sure that several PC software applications cannot access the C-887 at the same time. ➢ Make sure that you have selected the correct C-887 when establishing communication. ➢ If you cannot solve the problems, consult your network administrator if necessary.
	The starting procedure of the C-887 has not finished yet	1. Wait approx. 40 seconds after switching on or rebooting the C-887 (second signal tone was emitted and the **PWR** and **STA** LEDs light up).

Problem	Possible causes	Solution
		2. Try again to establish communication or send commands. When the IP addresses of the network devices are configured with AutoIP, it will take up to 2 minutes after the end of the starting procedure of the C-887 (p. 71) until communication is possible via TCP/IP.
	Another program is accessing the interface	➢ Close the other program.
	Problems with special PC software	➢ Check whether the system works with different PC software, such as for example, a terminal program or a development environment. You can test the communication by starting a terminal program (such as for example, PI Terminal) and entering `*IDN?` or `HLP?` ➢ Make sure that you end the commands with an LF (line feed). A command is only executed when LF has been received.

Problem	Possible causes	Solution
The customer software does not run with the PI drivers	Incorrect combination of driver routines/Vis	➢ Check whether the system works with a terminal program. If so: ➢ Read the information in the corresponding manual for the PC software and compare the sample code on the PI software CD with your program code.

If the problem that occurred with your system is not listed in the table above or cannot be solved as described, contact our customer service department ((p. 343)).

12 Customer Service Department

For inquiries and orders, contact your PI sales engineer or send us an email (mailto:service@pi.de).

- If you have questions concerning your system, provide the following information:
  - Product and serial numbers of all products in the system
  - Firmware version of the controller (if applicable)
  - Version of the driver or the software (if applicable)
  - PC operating system (if applicable)
- If possible: Take photographs or make videos of your system that can be sent to our customer service department if requested.

The latest versions of the user manuals are available for download on our website (p. 7).

13 Technical Data

In this Chapter

Subject to change. You can find the latest product specifications on the product web page at www.pi.ws (https://www.pi.ws).

13.1 Specifications

13.1.1 Data Table

	C-887.52 / C-887.521 / C-887.522 / C-887.523
Function	6-axis controller for hexapods, incl. control of two additional single axes Compact benchtop device Extending the functionality of C-887.52: C-887.521: Additional analog inputs C-887.522: Additional motion stop C-887.523: Additional motion stop and analog inputs
Drive type	Servo motors (hexapod and single axes)
Motion and servo controller	
Controller type	32-bit PID controller
Trajectory profiles	Jerk-controlled generation of dynamics profile with linear interpolation
Processor	Intel Atom dual core (1.8 GHz)
Servo cycle time	100 µs
Encoder input	AB (quadrature) differential TTL signal, 50 MHz BiSS
Stall detection	Servo off, triggered by position error
Reference switch	TTL

	C-887.52 / C-887.521 / C-887.522 / C-887.523
Electrical properties	
Hexapod control	12-bit PWM signal, TTL, 24 kHz
Hexapod power supply	24 V
Maximum output current	7 A
Interfaces and operation	
Communication interfaces	TCP/IP, RS-232 USB (HID, manual control unit)
Hexapod connection	HD D-sub 78 (f) for data transmission M12 4 (f) for power supply
Connectors for single axes	D-sub 15 (f)
I/O lines	HD D-sub 26 (f): 4 × analog input (-10 to 10 V, via 12-bit A/D converter) 4 × digital input (TTL) 4 × digital output (TTL)
Analog inputs	C-887.521 and C-887.523 only: 2 × BNC, -5 V to 5 V, via 16-bit A/D converter, 5 kHz bandwidth
Input for motion stop	C-887.522 and C-887.523 only: M12 8 (f)
Command set	PI General Command Set (GCS)
User software	PIMikroMove
Application programming interfaces	API for C / C++ / C# / VB.NET / MATLAB / Python, drivers for NI LabVIEW
Manual control	Optional: C-887.MC2 manual control unit for hexapods
Miscellaneous	
Operating voltage	24 V (external power adapter for 100 to 240 V AC, 50 / 60 Hz included in the scope of delivery)
Maximum current consumption	8 A
Operating temperature range	5 to 40 °C
Mass	2.8 kg
Dimensions	280 (320) mm × 150 mm × 103 mm

	C-887.53 / C-887.531 / C-887.532 / C-887.533
Function	6-axis controller for hexapods, incl. control of two additional single axes Compact benchtop device with EtherCAT interface Extending the functionality of C-887.53: C-887.531: Additional analog inputs C-887.532: Additional motion stop C-887.533: Additional motion stop and analog inputs
Drive type	DC motors (hexapod and single axes)

	C-887.53 / C-887.531 / C-887.532 / C-887.533
EtherCAT Specifications	
Fieldbus protocol	EtherCAT (CoE = CANopen over EtherCAT)
Drive profile	CiA402 drive profile (IEC 61800-7-201)
Cycle time	≥1 ms
Supported modes of operation	Reference move (homing mode) Positioning mode with cyclic target position via the PLC (cyclic synchronous position mode) Safe basic state for activating coordinate systems
Supported synchronization modes	Distributed clocks (DC), synchronous with SYNC0 event
Motion and servo controller	
Controller type	32-bit PID controller
Trajectory profiles	Jerk-controlled generation of dynamics profile with linear interpolation
Processor	Intel Atom dual core (1.8 GHz)
Servo cycle time	100 µs
Encoder input	AB (quadrature) differential TTL signal, 50 MHz BiSS
Stall detection	Servo off, triggered by position error
Reference switch	TTL
Electrical properties	
Hexapod control	12-bit PWM signal, TTL, 24 kHz
Hexapod power supply	24 V
Maximum output current	7 A
Interfaces and operation	
Communication interfaces	2 x RJ45 for EtherCAT (In/Out) TCP/IP, RS-232 USB (HID, manual control unit)
Hexapod connection	HD D-sub 78 (f) for data transmission M12 4 (f) for power supply
Connectors for single axes	D-sub 15 (f)
I/O lines	HD D-sub 26 (f): 4 × analog input (-10 to 10 V, via 12-bit A/D converter) 4 × digital input (TTL) 4 × digital output (TTL)
Analog inputs	C.887.531 and C-887.533 only: 2 x BNC, -5 V to 5 V, via 16-bit A/D converter, 5 kHz bandwidth
Input for motion stop	C-887.532 and C-887.533 only: M12 8 (f)
Command set	PI General Command Set (GCS)

	C-887.53 / C-887.531 / C-887.532 / C-887.533
User software	PIMikroMove
Application programming interfaces	API for C / C++ / C# / VB.NET / MATLAB / Python, drivers for NI LabVIEW
Manual operation	Optional: C-887.MC2 control unit for hexapods
Miscellaneous	
Operating voltage	24 V (external power adapter for 100 to 240 V AC, 50 / 60 Hz included in the scope of delivery)
Maximum current consumption	8 A
Operating temperature range	5 to 40 °C
Mass	2.8 kg
Dimensions	280 (320) mm × 150 mm × 103 mm

13.1.2 Specifications of the Analog Inputs

Connector	I/O **socket**	**BNC sockets Analog In 5 and Analog In 6**
Availability	All C-887.5xx models	C-887.521, .523, .531, .533 models only
Input voltage range	-10 to 10 V	-5 to 5 V
Max. permissible overvoltage	12.5 V	25 V
ADC resolution	12-bit	16-bit
Absolute accuracy	10-bit	13-bit
Sampling rate	10 kHz	10 kHz
Bandwidth	100 Hz	5 kHz
Input impedance	15 kΩ	15 kΩ
Channel identifiers for use in commands	1 to 4 (identifiers correspond to the numbers in the signal names; see "I/O Connection" (p. 354))	**Analog In 5**: 5 **Analog In 6**: 6

13.1.3 Cycle Times

Axis	**X, Y, Z, U, V, W**	**A, B, hexapod struts 1 to 6**
Cycle time	1 ms	0.1 ms

The differences in the cycle times result from the drive concepts:

- Axes X, Y, Z, U, V and W of the motion platform of the hexapod: Motion results from the motion of hexapod struts 1 to 6
- Hexapod struts 1 to 6 and axes A and B: Closed-loop drives with PWM amplifiers

13.1.4 Maximum Ratings

The C-887 is designed for the following operating data:

Input on:	**Maximum operating voltage** ⚠	**Maximum operating frequency** ⚠	**Maximum current consumption** ⚠
M12 4-pin panel plug (m)	24 V	⎓	8 A

Output on:	**Maximum output voltage** ⚠	**Maximum output frequency** ⚠	**Maximum output current** ⚠
M12 4-pole socket (f)	24 V	⎓	7 A

13.1.5 Ambient Conditions and Classifications

The following ambient conditions and classifications for the C-887 must be observed:

Area of application	For indoor use only
Maximum altitude	2000 m
Air pressure	1100 hPa to 0.1 hPa
Relative humidity	Highest relative humidity 80 % for temperatures up to 31 °C Decreasing linearly to 50 % relative air humidity at 40 °C
Storage temperature	0 °C to 70 °C
Transport temperature	–25 °C to +85 °C
Overvoltage category	II
Protection class	I
Degree of pollution	2
Degree of protection according to IEC 60529	IP20

13.2 System Requirements

The following system requirements must be met to operate the hexapod system:

- Suitable Hexapod from PI
- Suitable cable set from PI
- C-887 with power adapter
- PC with at least 30 MB of free memory and one of the following operating systems:
  - Windows: Versions 8.1, 10, 11 (32-bit, 64-bit)
  - Linux
- Communication interface to the PC:
  Unused COM port on the PC
  or
  Ethernet connection in the PC or an access point in the network connected to the PC via TCP/IP
- RS-232 or network cable to connect the C-887 with the PC or with the network
- PI software CD

13.3 Dimensions

Dimensions in mm. Note that the decimal points are separated by a comma in the drawings.

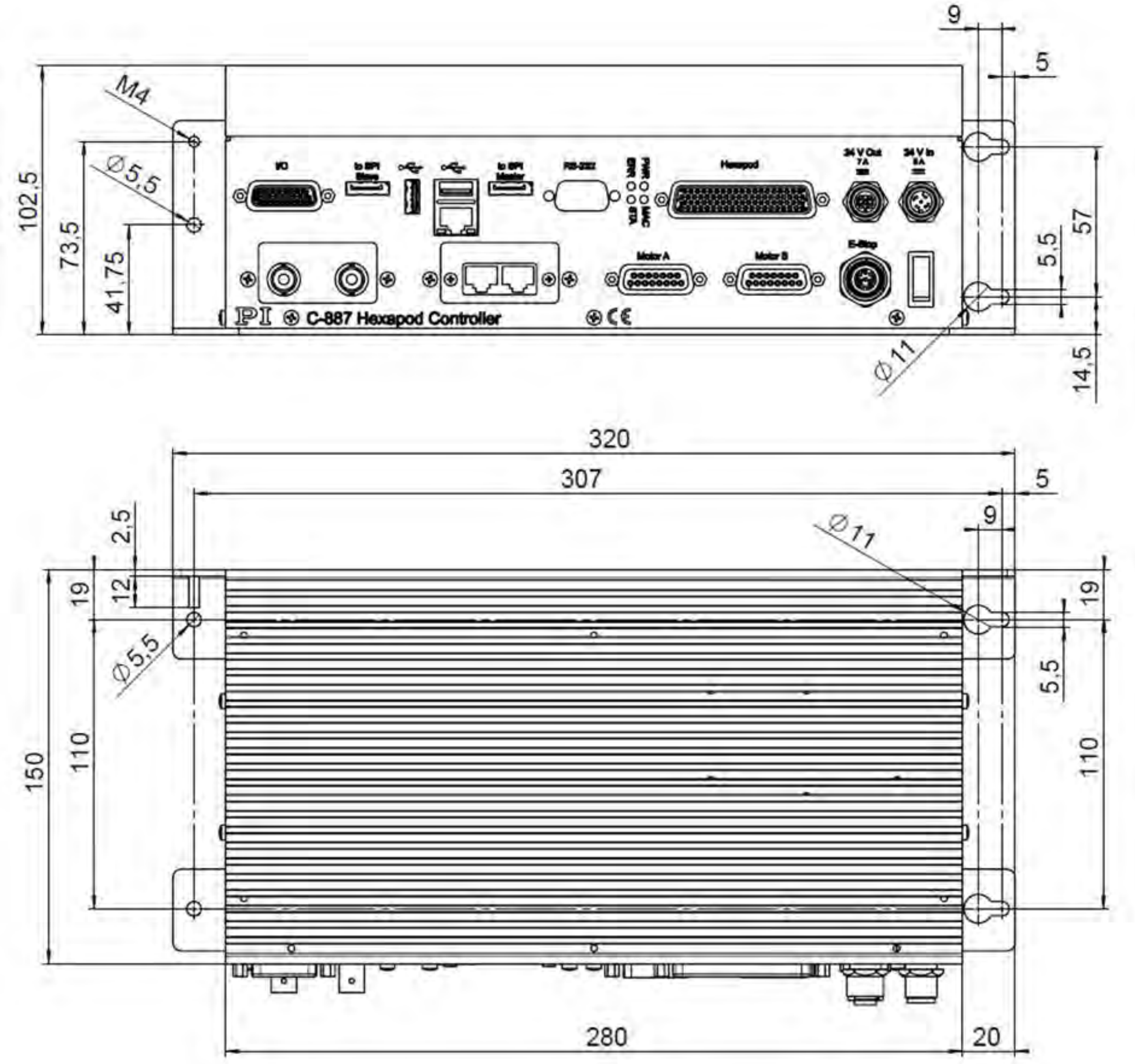


Figure 15: Dimensions valid for all C-887.5xx models

13.4 Cable Specifications

13.4.1 Data Transmission and Power Supply Cables

Data transmission cable	Power supply cable, single-side angled connector	Power supply cable, straight connectors
Alle Hexapodtypen	H-820, H-824, H-825, H-840, H-850	H-810, H-811, H-206
C-815.82D02	C-815.82P02A	C-815.82P02E
C-815.82D03	C-815.82P03A	C-815.82P03E
C-815.82D05	C-815.82P05A	C-815.82P05E
C-815.82D07	C-815.82P07A	C-815.82P07E
C-815.82D10	C-815.82P10A	C-815.82P10E
C-815.82D20	C-815.82P20A	C-815.82P20E

The models differ with respect to the following features:

1. Cable type
2. Length
3. Connector type (power cables only)

These features are coded in the product number by the character after the C-815.82 as follows:

Character following the C-815.82	Meaning	Possible values
First character	Cable type	D – Data transmission cable P – Power supply cable
Second character	Length	02 – 2 m 03 – 3 m 05 – 5 m 07 – 7.5 m 10 – 10 m 20 – 20 m
Third character	Connector type (power supply cable only)	A – Angled connector E – Straight connector

General		Unit
Cable length L	2 / 3 / 5 / 7.5 / 10 / 20	m
Maximum velocity	3	m/s
Maximum acceleration	5	m/s²
Maximum number of bending cycles	1 million	
Operating temperature range	-10 to +70	°C

Power supply cable, straight connectors		Unit
Minimum bending radius in a drag chain	49	mm
Minimum bending radius with the fixed installation	25	mm
Outer diameter	4.9	mm
Connectors	M12 m/f	

Power supply cable, angled connector			Unit
Cable length L	3	2 / 5 / 7.5 / 10 / 20	m
Minimum bending radius in a drag chain	72	94	mm
Minimum bending radius with the fixed installation	36	57	mm
Outer diameter	7.2	7.5	mm
Connector	M12 m/f		

Data transmission cable		Unit
Minimum bending radius in a drag chain	107	mm
Minimum bending radius with the fixed installation	81	mm
Outer diameter	10.7	mm
Connectors	HD D-sub 78 m/f	

13.5 Pin Assignment

13.5.1 Power Supply Connection

Power supply via 4-pin M12 panel plug (m) **24 V In 8 A**

Pin	Signal	
1	GND	
2	GND	
3	24 V DC	
4	24 V DC	

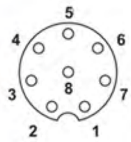
13.5.2 Supply Voltage for the Hexapod

Output of the supply voltage for the drives of the hexapod via 4-pole M12 socket (f)
24 V Out 7 A

Pin	Signal	
1	GND	
2	GND	
3	24 V DC	
4	24 V DC	

13.5.3 E-Stop

8-pole M12 socket (f)

Pin	Signal	
1	Break contact 1	 
2	Break contact 2	
3	Reserved	
4	Reserved	
5	Make contact 1	
6	Make contact 2	
7	Reserved	
8	Reserved	

13.5.4 I/O Connection

HD D-sub 26 socket (f)

Pin	Pin	Pin	Signal	
	10		Analog input 1	
1			Analog input 2	
		19	Analog input 3	
	11		Analog input 4	
2			GND (analog)	
		20	GND	
	12		Reserved	
3			Reserved	
		21	Reserved	

Pin	Pin	Pin	Signal	
	13		Reserved	
4			Reserved	
		22	GND	
	14		Reserved	
5			Reserved	
		23	Reserved	
	15		Reserved	
6			Vcc (+5 V, max. 500 mA)	
		24	GND	
	16		Digital input 4 (TTL)	
7			Digital input 3 (TTL)	
		25	Digital input 2 (TTL)	
	17		Digital input 1 (TTL)	
8			Digital output 4 (TTL)	
		26	Digital output 3 (TTL)	
	18		Digital output 2 (TTL)	
9			Digital output 1 (TTL)	

Analog inputs: -10 V to 10 V, 12-bit; 15 kΩ input impedance

Digital outputs:

Rise time and fall time = max. 500 ns

Output current = max. 10 mA per pin

Digital inputs:

Input impedance = 10 kΩ

Input voltage = 0 to 5.5 V

Schmitt trigger input

	min	max
V_{T+} (switching threshold with rising input voltage)	1.3 V	2.2 V
V_{T-} (switching threshold with falling input voltage)	0.6 V	1.3 V
ΔV_T (Hysteresis; V_{T+} - V_{T-})	0.4 V	1.1 V

13.5.5 Hexapod

Data transmission between the hexapod and controller, power supply for the sensors of the hexapod

HD D-sub 78 socket (f)

Signal type	Socket
All signals: TTL	

Pin Assignment

Pin	Pin	Signal
1		CH1 Sign OUT / MO-
	21	CH1 Ref IN
2		nc
	22	CH1 A+ IN / MA+
3		CH1 A- IN / MA-
	23	GND
4		CH2 Sign OUT / MO-
	24	CH2 Ref IN
5		nc
	25	CH2 A+ IN / MA+
6		CH2 A- IN / MA-
	26	GND
7		CH3 Sign OUT / MO-
	27	CH3 Ref IN
8		nc
	28	CH3 A+ IN / MA+
9		CH3 A- IN / MA-
	29	GND
10		CH4 Sign OUT / MO-
	30	CH4 Ref IN
11		nc
	31	CH4 A+ IN / MA+

Pin	Pin	Signal
40		CH1 MAGN OUT / MO+
	60	CH1 LimP IN
41		CH1 LimN IN
	61	CH1 B+ IN / SL+
42		CH1 B- IN / SL-
	62	GND
43		CH2 MAGN OUT / MO+
	63	CH2 LimP IN
44		CH2 LimN IN
	64	CH2 B+ IN / SL+
45		CH2 B- IN / SL-
	65	GND
46		CH3 MAGN OUT / MO+
	66	CH3 LimP IN
47		CH3 LimN IN
	67	CH3 B+ IN /SL+
48		CH3 B- IN / SL-
	68	GND
49		CH4 MAGN OUT / MO+
	69	CH4 LimP IN
50		CH4 LimN IN
	70	CH4 B+ IN / SL+

Pin	Pin	Signal
12		CH4 A- IN / MA-
	32	GND
13		CH5 Sign OUT / MO-
	33	CH5 Ref IN
14		nc
	34	CH5 A+ IN / MA+
15		CH5 A- IN / MA-
	35	GND
16		CH6 Sign OUT / MO-
	36	CH6 Ref IN
17		nc
	37	CH6 A+ IN / MA+
18		CH6 A- IN / MA-
	38	GND
19		ID chip
	39	GND
20		24 V output

Pin	Pin	Signal
51		CH4 B- IN / SL-
	71	GND
52		CH5 MAGN OUT / MO+
	72	CH5 LimP IN
53		CH5 LimN IN
	73	CH5 B+ IN / SL+
54		CH5 B- IN / SL-
	74	GND
55		CH6 MAGN OUT / MO+
	75	CH6 LimP IN
56		CH6 LimN IN
	76	CH6 B+ IN / SL+
57		CH6 B- IN / SL-
	77	GND
58		Brake/Enable drive
	78	GND
59		Power Good input*

* Return of the current operating voltage of the hexapod. The drives of the hexapod are being properly supplied with power when the Power Good signal is the required operating voltage of the hexapod (tolerance range ±10 %). The Power Good signal can be checked with the `DIA?` command (p. 154). Refer to "Protective Functions of the C-887" (p. 88) for further information.

As of serial number 121017873, the C-887 supports the MA, MO, and SL signals, which are used for data transmission between the hexapod and the controller via the BiSS protocol.

13.5.6 Motor A, Motor B

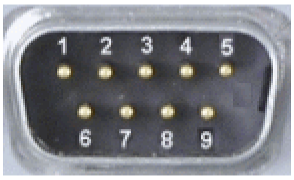
D-sub 15 socket, female

Pin	Pin	Signal	
1		Output: +5 V for motor brake	
	9	Reserved	
2		Reserved	
	10	PWM GND	
3		Output: MAGN (motor PWM, TTL signal)	

Pin	Pin	Signal	
	11	Output: SIGN (direction of rotation of the motor, TTL signal)	
4		Output: +5 V, for encoder	
	12	Input: Negative limit switch	
5		Input: Positive limit switch	
	13	Input: REFS (reference switch, TTL signal)	
6		Reserved	
	14	Input: encoder: A (+) / ENCA	
7		Input: encoder: A (-)	
	15	Input: encoder: B (+) / ENCB	
8		Input: encoder: B (-)	

13.5.7 RS-232

D-sub 9 panel plug, male

Pin	Function	
1	Not connected	
2	RxD, data input	
3	TxD, data output	
4	Not connected	
5	DGND (digital mass)	
6	Not connected	
7	RTS, hardware handshake output	
8	CTS, hardware handshake input	
9	Not connected	

13.6 Status Registers

13.6.1 Status Registers for Hexapod Struts and Axes A and B

The C-887 has one status register for each of struts 1 to 6 of the hexapod and for axes A and B. You can query the bits of these registers with the SRG? command (p. 247) and record with the C-887's data recorder (p. 97), record option 80 (status register of axis).

Bit	15	14	13	12	11	10	9	8
Description	On target	Reference move is executed	In motion	Servo mode on	-	-	-	Error flag

Bit	7	6	5	4	3	2	1	0
Description	-	-	-	-	-	Pos. limit switch (when evaluation is activated*)	Reference switch	Neg. limit switch (when evaluation is activated*)

* The limit switch evaluation is activated with the ***Has No Limit Switches*** parameter (ID 0x32). Activation is only effective when the mechanics actually supply the limit switch signals to the C-887.

Unassigned bits have the value 0.

The state of bits 13 to 15 is based on the criteria for determining the motion status (p. 42).

13.6.2 System Status Register

You can query the bits of the following register with the STA? (p. 251) and #4 (p. 143) commands:

Bit:	23	22	21	20	19	18	17	16
	-	-	-	-	Reference move is executed	Reference move for axis B successful	Reference move for axis A successful	Reference move for hexapod successful
Bit:	15	14	13	12	11	10	9	8
	Axis B in motion	Axis A in motion	Strut 6 in motion	Strut 5 in motion	Strut 4 in motion	Strut 3 in motion	Strut 2 in motion	Strut 1 in motion
Bit:	7	6	5	4	3	2	1	0
	Motion error axis B	Motion error axis A	Motion error strut 6	Motion error strut 5	Motion error strut 4	Motion error strut 3	Motion error strut 2	Motion error strut 1

Unassigned bits have the value 0.

The state of bits 8 to 15 is based on the criteria for determining the motion status (p. 42).

Example:

Send: STA?

Receive: 0x71804

The response is given in hexadecimal format. This means: A motion error was reported for strut 3; struts 4 and 5 are in motion. The reference move of the hexapod and axes A and B was completed successfully.

14 Old Equipment Disposal

In accordance with EU law, electrical and electronic equipment may not be disposed of in EU member states via the municipal residual waste.

Dispose of your old equipment according to international, national, and local rules and regulations.

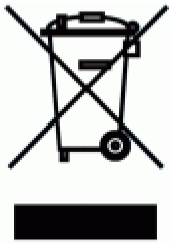
To fulfill the responsibility as the product manufacturer, Physik Instrumente (PI) GmbH & Co. KG undertakes environmentally correct disposal of all old PI equipment made available on the market after 13 August 2005 without charge.

Any old PI equipment can be sent free of charge to the following address:

Physik Instrumente (PI) GmbH & Co. KG

Auf der Roemerstrasse 1

76228 Karlsruhe, Germany



15 European Declarations of Conformity

For the C-887, declarations of conformity were issued according to the following European statutory requirements:

EMC Directive

RoHS Directive

The standards applied for certifying conformity are listed below.

EMC: EN 61326-1

Safety: EN 61010-1

RoHS: EN IEC 63000