

fMS232E  
**C-877 PLine® Controller**  
User Manual

Version: 2.0.0

Date: 10/24/2024



**This document describes the following product:**

- **C-877.1U11**  
Compact, inexpensive piezo motor controller / driver, 1 axis, for PLine® systems with low power consumption



The following trademarks are the intellectual property of Physik Instrumente (PI) SE & Co. KG ("PI") and have been entered in the trademark register of the German Patent and Trade Mark Office and, in some cases, also in other trademark registers under the company name of Physik Instrumente (PI) GmbH & Co. KG: PI®, PIC®, PICMA®, PILine®, PIFOC®, PiezoWalk®, NEXACT®, NEXLINE®, PInano®, NanoCube®, Picoactuator®, PicoCube®, PIMikroMove®, PIMag®, PIHera®

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, FTDI

These designations are used for identification purposes only.

The software products provided by PI are subject to the General Software License Terms of Physik Instrumente (PI) SE & 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/fileadmin/user\\_upload/physik\\_instrumente/files/legal/General-Software-License-Agreement-Physik-Instrumente.pdf](https://www.physikinstrumente.com/fileadmin/user_upload/physik_instrumente/files/legal/General-Software-License-Agreement-Physik-Instrumente.pdf)) and in the Third-Party Software Notes ([https://www.physikinstrumente.com/fileadmin/user\\_upload/physik\\_instrumente/files/legal/Third-Party-Software-Note-Physik-Instrumente.pdf](https://www.physikinstrumente.com/fileadmin/user_upload/physik_instrumente/files/legal/Third-Party-Software-Note-Physik-Instrumente.pdf)) on our website.

© 2024 Physik Instrumente (PI) SE & Co. KG, Karlsruhe, Germany. The text, photographs, and drawings in this manual are protected by copyright. Physik Instrumente (PI) SE & Co. KG reserves all rights in this respect. The use of any text, images and drawings is permitted only in part and only when indicating the source.

Original instructions

First printing: 10/24/2024

Document number: MS232E, ASt, Version 2.0.0

Subject to change. This manual is superseded by any new release. The latest respective release is available for download on our website (<https://www.physikinstrumente.com/en/>).

## Contents

<b>1</b>	<b>About this Document</b>	<b>1</b>
1.1	Objective and Target Audience of this User Manual.....	1
1.2	Symbols and Typographic Conventions.....	1
1.3	Definition of Terms.....	2
1.4	Figures .....	3
1.5	Other Applicable Documents .....	3
1.6	Downloading Manuals.....	3
<b>2</b>	<b>Safety</b>	<b>5</b>
2.1	Intended Use .....	5
2.2	General Safety Instructions .....	5
2.3	Organizational Measures.....	6
2.4	European Declarations of Conformity.....	6
<b>3</b>	<b>Product Description</b>	<b>7</b>
3.1	Product View .....	7
3.1.1	Front Panel .....	7
3.1.2	Rear Panel.....	8
3.1.3	Type Plate .....	8
3.2	Scope of Delivery.....	9
3.3	Overview of PC Software.....	9
3.3.1	PI Software Suite .....	9
3.4	Positioner Databases.....	11
3.5	ID Chip Detection.....	12
3.6	Communication Interfaces .....	12
3.7	Functional Principles .....	13
3.7.1	Block Diagram.....	13
3.7.2	Commandable Elements.....	13
3.7.3	Important Components of the Firmware .....	14
3.7.4	Operating Modes.....	15
3.7.5	Physical Units.....	16
3.7.6	Motion Triggering .....	17
3.7.7	Generation of the Dynamics Profile .....	18
3.7.8	Servo Algorithm and Other Control Value Corrections .....	20
3.7.9	Optional Two-Phase Control.....	26
3.7.10	On-Target State .....	27
3.7.11	Supported Motor Types.....	28
3.7.12	Automatic Frequency Control .....	28
3.7.13	Reference Switch Detection .....	29
3.7.14	Limit Switch Detection.....	30
3.7.15	Travel Range and Soft Limits .....	31
3.7.16	Referencing.....	34

<b>4</b>	<b>Unpacking</b>	<b>39</b>
<b>5</b>	<b>Installing</b>	<b>41</b>
5.1	General Notes on Installation.....	41
5.2	Mounting the C-877 .....	41
5.3	Connecting the C-877 to the Protective Earth Conductor .....	42
5.4	Connecting the Power Adapter to the C-877 .....	43
5.5	Connecting the Positioner .....	43
5.6	Installing the PC Software .....	44
5.6.1	Doing Initial Installation.....	44
5.6.2	Installing Updates .....	45
5.6.3	Installing a Custom Positioner Database .....	46
5.7	Connecting the PC .....	47
5.7.1	Connecting the C-877 via the USB interface .....	47
<b>6</b>	<b>Startup</b>	<b>49</b>
6.1	General Notes on Startup.....	49
6.2	Switching the C-877 On .....	49
6.3	Establishing Communication .....	50
6.3.1	Establishing Communication via the USB Interface .....	50
6.4	Starting Motion .....	51
6.5	Optimizing the Servo Control Parameters .....	56
<b>7</b>	<b>Operation</b>	<b>61</b>
7.1	Protective Functions of the C-877 .....	61
7.1.1	Protection Against Overheating .....	61
7.1.2	Behavior with Motion Errors .....	61
7.1.3	Behavior During System Errors.....	62
7.1.4	Re-establishing Readiness for Operation .....	62
7.2	Data Recorder.....	63
7.2.1	Configuring the Data Recorder .....	63
7.2.2	Starting the Recording.....	64
7.2.3	Reading Recorded Data .....	64
7.3	Controller Macros.....	64
7.3.1	Overview: Macro Functionality and Example Macros.....	64
7.3.2	Commands and Parameters for Macros.....	65
7.3.3	Working with Macros .....	66
7.3.4	Making Backups and Loading Controller Macros .....	72
<b>8</b>	<b>GCS Commands</b>	<b>75</b>
8.1	Notation.....	75
8.2	GCS Syntax for Syntax Version 2.0 .....	75
8.3	Target and Sender Address .....	77
8.4	Variables .....	78
8.5	Command Overview .....	79

8.6	Command Descriptions for GCS 2.0 .....	83
8.7	Error Codes.....	140
<b>9</b>	<b>Adapting Settings</b>	<b>163</b>
9.1	Settings of the C-877 .....	163
9.2	Changing Parameter Values in the C-877.....	163
9.2.1	General Commands for Parameters.....	164
9.2.2	Commands for Fast Access to Individual Parameters .....	164
9.2.3	Saving Parameter Values in a Text File.....	165
9.2.4	Changing Parameter Values: General Procedure .....	166
9.3	Creating or Changing a Positioner Type .....	168
9.4	Parameter Overview.....	171
<b>10</b>	<b>Maintenance</b>	<b>183</b>
10.1	Cleaning the C-877 .....	183
10.2	Updating Firmware.....	183
<b>11</b>	<b>Troubleshooting</b>	<b>187</b>
<b>12</b>	<b>Customer Service Department</b>	<b>191</b>
<b>13</b>	<b>Technical Data</b>	<b>193</b>
13.1	Specifications.....	193
13.1.1	Data Table.....	193
13.1.2	Maximum Ratings.....	194
13.1.3	Ambient Conditions and Classifications .....	194
13.2	Dimensions .....	195
13.3	Pin Assignment .....	196
13.3.1	Motor and Sensor .....	196
13.3.2	Power supply connection .....	196
<b>14</b>	<b>Old Equipment Disposal</b>	<b>197</b>



# 1 About this Document

## 1.1 Objective and Target Audience of this User Manual

This user manual contains the information required for using the C-877 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. 3).

## 1.2 Symbols and Typographic Conventions

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

### NOTICE



**Dangerous situation**  
Failure to comply could result in damage to the equipment.  
➤ Precautions to avoid the risk.

### INFORMATION

Information for easier handling, tricks, tips, etc.

Symbol/Label	Meaning
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.
<i>Start &gt; Settings</i>	Menu path in the PC software (example: to open the menu, the <b>Start</b> and <b>Settings</b> menu items must be selected successively)
POS?	Command line or a command from PI's General Command Set (GCS) (example: command to get the axis position)
<i>Device S/N</i>	Parameter name (example: parameter where the serial number is stored)
5	Value that must be entered or selected via the PC software

## 1.3 Definition of Terms

Term	Explanation
Axis	Also referred to as "logical axis". The logical axis represents the motion of the mechanics in the firmware of the C-877. For mechanics that allow motion in several directions (e.g., in X, Y, and Z), each direction of motion corresponds to a logical axis.
Positioner	Mechanics connected to the C-877. In the case of positioners with just one motion axis, the designation "axis" is synonymous with "positioner". Positioners that allow motion in several axes are also designated as "multi-axis positioners". For these positioners, a distinction must be made between the individual axes.
Control value	The control value is the input for the PILine® driver electronics of the C-877. The driver electronics convert the control value into the piezo voltage for the axis of the positioner.
Two-phase control	A PILine® piezo motor has a separate piezo segment for the positive and the negative direction of motion. Depending on the direction of motion, only the corresponding segment is normally driven by the piezo voltage. The two-phase control allows the second segment to be driven parallel to the first segment at specified time intervals, in order to interrupt the feed motion of the piezo motor for the duration of the interval. Depending on the application, interruption of the feed motion at intervals can improve the settling behavior of the axis.
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.
Dynamics profile	Comprises the target position, velocity, and acceleration of the axis calculated by the profile generator of the C-877 for any point in time of the motion. The calculated values are called "commanded values".
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.
Nonvolatile memory	Memory module (read-only memory, e.g., EEPROM or flash memory) from which the default values of the parameters are loaded into the volatile memory when the controller is started. In the PC software from PI, the parameter values in the nonvolatile memory are also referred to as "startup values".
Firmware	Software that is installed on the controller.
PC software	Software installed on the PC.
GCS	PI General Command Set: command set for PI controllers



## 1.4 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.

## 1.5 Other Applicable Documents

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

Description	Document
Short instructions for the installation and startup of the C-877.1U11	MS242EK Short Instructions for Digital Motor Controllers
PI MATLAB Driver GCS 2.0	SM155E Software Manual
PI GCS 2.0 DLL0	SM151E Software Manual
GCS array: Data format description	SM146E Software Manual
PIMikroMove	SM148E Software Manual
PIStageEditor: Software for the management of positioner databases	SM144E Software Manual
PI Update Finder: Search and download updates	A000T0028 User Manual
PI Software on ARM-Based Platforms	A000T0089 Technical Note
Downloading manuals from PI: PDF file with links to the manuals for digital electronics and software from PI. Supplied with the PI software.	A000T0081 Technical Note

The latest versions of the user manuals are available for download on our website (p. 3).

## 1.6 Downloading Manuals

### INFORMATION

If a manual is missing or problems occur with downloading:

- Contact our customer service department (p. 191).

### Downloading manuals

1. Open the website **www.pi.ws**.
2. Search the website for the product number (e.g., C-877).
3. In the search results, select the product to open the product detail page.
4. Select **Downloads**.

The manuals are shown under **Documentation**. Software manuals are shown under **General Software Documentation**.

5. For the desired manual, select **ADD TO LIST** and then **REQUEST**.
6. Fill out the request form and select **SEND REQUEST**.

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

## 2 Safety

### 2.1 Intended Use

The C-877 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.

In accordance with its design, the C-877 is intended for the operation of positioners with PLine® ultrasonic piezomotors and Sub-D 15 (m) connectors.

The C-877 is intended for closed-loop operation with incremental position sensors. In addition, it can read and process the reference point and limit switch signals from the positioner connected.

The C-877 may only be used in compliance with the technical specifications and instructions in this user manual. The user is responsible for process validation.

The C-877 must not be used for purposes other than those named in this user manual. In particular, the C-877 must not be used to drive ohmic or inductive loads.

### 2.2 General Safety Instructions

The C-877 is built according to state-of-the-art technology and recognized safety standards. Improper use of the C-877 may result in personal injury and/or damage to the C-877.

- Use the C-877 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-877 correctly.

- Install the C-877 near the power source so that the power plug can be quickly and easily disconnected from the mains.
- Use the supplied components (power supply, adapter, power cord) to connect the C-877 to the power source.
- If one of the supplied components for connecting to the power source has to be replaced, use a sufficiently dimensioned component.

## 2.3 Organizational Measures

### User manual

- Always keep this user manual together with the C-877. The latest versions of the user manuals are available for download on our website (p. 3).
- Add all information from the manufacturer such as supplements or technical notes to the user manual.
- If you give the C-877 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-877 only after you have read and understood this user manual.

### Personnel qualification

The C-877 may only be installed, started, operated, maintained, and cleaned by authorized and appropriately qualified personnel.

## 2.4 European Declarations of Conformity

For the C-877, 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





## 3 Product Description

### 3.1 Product View

#### 3.1.1 Front Panel



Figure 1: Front panel of the C-877.1U11

Labeling	Type	Function
24 V  0.8 A	Barrel connector socket (p. 196)	Connection for the supply voltage
STA	LED green	Controller state: <ul style="list-style-type: none"> <li>On: C-877 is ready for normal operation</li> <li>Off: C-877 is not connected to the supply voltage or is in firmware update mode</li> </ul>
ERR	LED red	Error indicator: <ul style="list-style-type: none"> <li>On: Error (error code <math>\neq</math> 0)</li> <li>Off: No error (error code = 0)</li> </ul> The error code can be queried with the <b>ERR?</b> command. The query resets the error code to zero and the LED is switched off.
 	Threaded bolt with fastening material for protective earth conductor	Protective earth connection (p. 42) The threaded bolt must be connected to a protective earth conductor, because the C-877 is not grounded via the power supply connector.
	Mini-USB type B	Universal serial bus for connection to the PC

### 3.1.2 Rear Panel

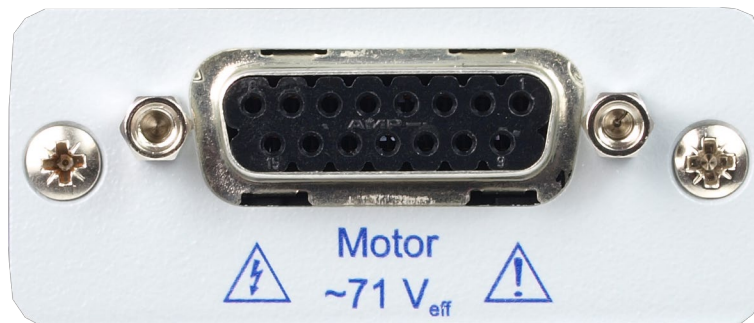







Figure 2: Rear panel of the C-877.1U11

Labeling	Type	Function
<b>Motor</b>  ~71 V <sub>eff</sub> 	Sub-D 15 (f) (p. 196)	Connector for positioner. Only for PILine® ultrasonic piezomotors! <ul style="list-style-type: none"> <li>▪ Outputs for piezo voltage</li> <li>▪ Input of the signals of the position sensor</li> <li>▪ Input of the signals from the limit switches and reference switch</li> <li>▪ Output of the supply voltage for position sensor, reference point and limit switches</li> <li>▪ Input for signals of the ID chip</li> </ul>

### 3.1.3 Type Plate

Labeling	Function
	Data matrix code (example; contains the serial number)
<b>C-877.1U11</b>	Product name
<b>PI</b>	Manufacturer's logo
<b>116056789</b>	Serial number (example), individual for each C-877 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. 197)
<b>CE</b>	CE conformity mark
<b>WWW.PI.WS</b>	Manufacturer's address (website)

## 3.2 Scope of Delivery

Article	Component
C-877.1U11	PILine® piezomotor controller
C-501.24050H	24 V 50 W wide-range-input power supply, barrel connector
3763	Power cord
000036360	USB cable (type A to Mini-B) for connection to the PC
C-990.CD1	Data storage device with PC software from PI
MS242EK	Short instructions for digital motor controllers

## 3.3 Overview of PC Software

### 3.3.1 PI Software Suite

A data storage device with the PI Software Suite is included in the C-877's scope of delivery (p. 9). Some components of the PI Software Suite are described in the table below. For information on the compatibility of the software with PC operating systems see the C-990.CD1 Release News in the root directory of the data storage device.

#### Libraries, drivers

PC software	Short description	Recommended use
Dynamic program library for GCS	Allows software programming for the C-877 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 NI LabVIEW drivers.
Drivers for use with NI LabVIEW software	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 controllers. The drivers support the PI GCS.	For users who want to use NI LabVIEW to program their application.
MATLAB drivers	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	For users who want to use MATLAB to program their application.

PC software	Short description	Recommended use
	script. This class supports the PI GCS. The PI MATLAB driver does not require any additional MATLAB toolboxes.	
USB driver	Driver for the USB interface	For users who want to connect the controller to the PC via the USB interface.

### User software

PC software	Short description	Recommended use
PIMikroMove	Graphic user interface for Windows with which the C-877 and other controllers from PI can be used. <ul style="list-style-type: none"> <li>▪ The system can be started without programming effort</li> <li>▪ Graph of motions in open-loop and closed-loop operation</li> <li>▪ Macro functionality for storing command sequences on the PC (host macros)</li> <li>▪ Support of HID devices</li> <li>▪ Complete environment for command entry, for trying out different commands</li> </ul> PIMikroMove uses the dynamic program library to supply commands to the controller.	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.
PITerminal	Terminal program that can be used for nearly all PI controllers.	For users who want to send GCS commands directly to the controller.
PIStages3Editor	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	Checks the PI software installed on the PC. If more current versions of the PC software are available on the PI server, downloading is offered.	For users who want to update the PC software.
PIFirmwareManager	Program for user support when updating firmware of the C-877.	For users who want to update the firmware.



### 3.4 Positioner Databases

You can select a suitable parameter set for your positioner from a positioner database in the PC software from PI. The PC software transfers the values of the selected parameter set to the volatile memory of the controller.

Database file name	Editable?	Description
PIMicosStages2.dat	No, updates can be downloaded from the PI website (p. 45).	Standard positioner database: Includes parameter sets for all standard positioners from PI miCos; is automatically saved to the PC when the PC software is installed.
PIStages2.dat	No, updates can be downloaded from the PI website (p. 45).	Standard positioner database: Includes parameter sets for all standard positioners from PI; is automatically saved to the PC when the PC software is installed.
PI_UserStages2.dat	Yes, new parameter sets can be created, edited, and saved (p. 168).	Is automatically created when you make a connection to your positioner for the first time using the PC software (i.e., when selecting the positioner in PIMikroMove or when using the commands <code>VST?</code> or <code>CST</code> from the dynamic program library).
X-xxx.dat	No, you receive updates from our customer service department (p. 191).	Contains the parameter set for a custom positioner, for installation see "Installing a Custom Positioner Database" (p. 46).

The parameter values in the volatile memory of the C-877 can be queried and written to the nonvolatile memory; see "Adapting Settings" (p. 163)

#### INFORMATION

Positioners only contain some of the information that is required to operate a positioner with the C-877. Further information is loaded as parameter values to the volatile memory of the C-877 from the ID chip (p. 11) of the positioner when the C-877 is switched on or rebooted.

Parameters that are loaded from a stage database or from the ID chip are marked in color in the parameter overview (p. 171).

## 3.5 ID Chip Detection

Positioners with PLine® ultrasonic piezomotors and Sub-D 15 connectors have an ID chip in the connector on which the following data is saved as parameters:

- Information on the positioner: Type, serial number, date of manufacture, version of the hardware

The data of the connected positioner is loaded from the ID-Chip into the volatile memory of the C-877 when the C-877 is switched on (p. 49) or rebooted.

The parameter values in the volatile memory of the C-877 can be queried and written to the nonvolatile memory, see "Adapting Settings" (p. 163).

### **INFORMATION**

The ID chip only contains some of the information that is required to operate the positioner with the C-877. When you use the PC software from PI, further information is loaded as parameter values from a positioner database (p. 11) into the volatile memory of the C-877.

Parameters that are loaded from the ID chip or from a positioner database are marked in color in the parameter overview (p. 171).

## 3.6 Communication Interfaces

The C-877 can be controlled with ASCII commands from a PC. Connection to the PC takes place via the USB interface of the C-877.

### **INFORMATION**

A USB UART module (FTDI) is used for the USB interface in the C-877. Therefore, if the C-877 is connected via USB and switched on, the USB interface is also shown as COM port in the PC software. The C-877 uses a baud rate of 115200 for this interface.

## 3.7 Functional Principles

### 3.7.1 Block Diagram

The C-877 controls the motion of the logical axis of a positioner. The following block diagram shows how the C-877 generates the piezo voltage for the axis connected:

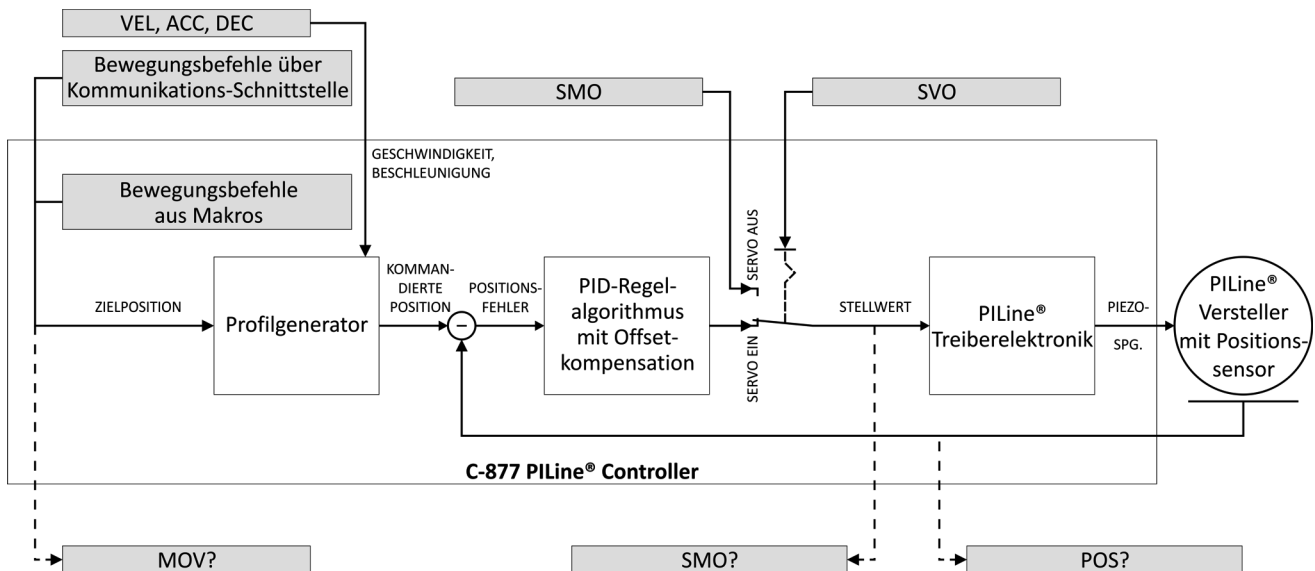


Figure 3: C-877: Control value generation

The C-877 supports positioners with PILine® ultrasonic piezomotor and incremental position sensor.

### 3.7.2 Commandable Elements

The following table contains the items that can be accessed with GCS commands (p. 82).

Item	Num-ber	Identifier	Description
Logical axis	1	1 (can be changed)	<p>The logical axis represents the motion of the positioner in the firmware of the C-877. It corresponds to the axis of a linear coordinate system.</p> <p>Motion for logical axes are commanded in the firmware of the C-877 (i.e., for the directions of motion of a positioner). The motion commands <b>MOV</b> and <b>MVR</b> are for example, available in closed-loop operation. Motion in open-loop operation is triggered by <b>SMO</b>.</p> <p>The axis identifier can be queried with the <b>SAI?</b> command and modified with the <b>SAI</b> command. It can consist of 8 characters; valid characters (display with the</p>

Item	Number	Identifier	Description
			<p>TVI? command) are: 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ_ If the <b>Stage Name</b> parameter (0x3C) has the value NOSTAGE, the axis is "deactivated". 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.</p>
Data recorder tables	4	1 to 4	The C-877 has 4 data recorder tables (queried with TNR?) with 1024 data points per table.
Overall system	1	1 (cannot be changed)	C-877 as an overall system

### 3.7.3 Important Components of the Firmware

The firmware of the C-877 provides the following functional units:

Firmware component	Description
Parameters	<p>Parameters reflect the properties of the positioner connected (e.g., travel range) and specify the behavior of the C-877 (e.g., settings for the servo algorithm).</p> <p>The parameters can be divided into the following categories:</p> <ul style="list-style-type: none"> <li>Protected parameters whose default settings cannot be changed</li> <li>Parameters that must be set by the user to adapt to the application</li> </ul> <p>For further information, see "Adapting Settings" (p. 163).</p> <p>In the case of positioners with ID chip, the values of some parameters are stored on the ID chip. They are loaded to the volatile memory when switching on or rebooting the C-877.</p>
Command levels	<p>The command levels determine the write permission for the parameters. The current command level can be changed with the CCL command. This may require entering a password.</p>
ASCII commands (GCS)	<p>Communication with the C-877 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).</p> <p>Examples of the use of GCS:</p> <ul style="list-style-type: none"> <li>Configuring the C-877</li> <li>Setting the operating mode</li> <li>Starting motion of the positioner</li> <li>Getting system and position values</li> </ul> <p>You can find a list of the available commands in the "Command Overview" section (p. 79).</p>

Firmware component	Description
Profile generator and servo control algorithm	The profile generator does calculations to specify the target position, velocity, and acceleration of the axis for any point in time during motion. The position error resulting from the difference between the target position and the actual position (sensor feedback) runs through a PID servo algorithm. For further information, see "Generating a Dynamics Profile" (p. 18) and "Servo Algorithm and Other Control Value Corrections" (p. 20).
Data recorder	The C-877 contains a real-time data recorder (p. 63). The data recorder can record various signals (e.g., position) from different data sources (e.g., logical axis or input channels).
Macros	The C-877 can save macros (p. 64). Command sequences can be defined and stored permanently in the nonvolatile memory of the device via the macro function. A startup macro can be defined that runs each time the C-877 is switched on or rebooted. The startup macro simplifies stand-alone operation (operation without a connection to the PC). Further information can be found in the "Controller Macros" section (p. 64).

The firmware can be updated with a tool (p. 183).

### 3.7.4 Operating Modes

The C-877 supports the following operating modes:

Operating Mode	Description
Closed-loop operation (Servo mode on)	A profile generator calculates the dynamics profile from the values specified for target position, velocity, acceleration, and deceleration. The position error that results from the difference between the calculated dynamics profile and the actual position (sensor feedback) runs through a PID servo algorithm ( <b>proportional integral derivative</b> ). Additional corrections can be made as well. The result is the control value for the driver electronics integrated in the C-877. Further information can be found in the sections "Generation of Dynamics Profile" (p. 18) and "Servo Algorithm and Other Control Value Corrections" (p. 20).
Open-loop operation (Servo mode off)	In open-loop operation, the C-877 does not calculate a dynamics profile and does not evaluate the signals of the position sensor. As a result, the positioner can move unbraked to the end of the travel range and, despite the limit switch function, strike the hard stop.

#### INFORMATION

The C-877 is intended for closed-loop operation with position sensors (servo mode On). After switch-on, open-loop operation is active by default (servo mode Off).

- Query the current operating mode with the `SVO?`, `#4` or `SRG?` commands.
- Enable closed-loop operation with the `SVO` command.
- If necessary, program a startup macro that starts the C-877 via the `SVO` command in closed-loop operation; see "Setting up a startup macro" (p. 71).
- Avoid motion in open-loop operation.

### 3.7.5 Physical Units

The C-877 supports various units of length for positions. Adapting is done by a factor that converts the incremental encoder counts into the physical unit of length required. The conversion factor is set with the following parameters:

Parameters	Description and Possible Values
<b>Numerator Of The Counts-Per-Physical-Unit Factor</b> 0xE	Numerator and denominator of the factor for counts per physical length unit 1 to 1.000.000.000 for each parameter. The factor for the counts per physical unit of length specifies the unit of length for position queries and motion commands in closed-loop operation. The values of every parameter, whose unit is either the physical unit of length itself or a unit of measurement based on it, are automatically adapted to the set factor. The factor for the counts per physical unit of length has no impact on the stability of the servo loop but is used for the input and output scaling of position values.
<b>Denominator Of The Counts-Per-Physical-Unit Factor</b> 0xF	

The unit symbol can be customized for display purposes with the following parameter:

Parameters	Description and Possible Values
<b>Axis Unit</b> 0x07000601	Unit symbol Maximum of 20 characters. For example, the unit symbol is "MM", if the factor for the counts per physical unit of length is set with parameters 0xE and 0xF so that the encoder counts are converted into millimeters. The unit for rotation stages is normally "deg". The value of the parameter 0x07000601 is not evaluated by the C-877 but is used by the PC software for display purposes. Examples: 1 encoder count = 100 nm Counts per physical length unit: 10000:1 → Unit symbol: mm 1 encoder count = 0.254 mm Counts per physical length unit: 100:1 → Unit symbol: inch

### 3.7.6 Motion Triggering

#### Motion in closed-loop operation

The following table is valid for motion in closed-loop operation.

Trigger of the motion	Commands	Description
Motion commands, sent from the command line or via the PC software	MOV, MVR	Motion to absolute or relative target position
	GOH	Motion to zero position
	STE	Starts performing a step and records the step response
	FNL, FPL, FRF	Starts reference moves
	FED	Starts moves to signal edges
Controller macros with motion commands	MAC	Calls a macro function. Permits recording, deleting and running macros on the controller. Any commands can be sent from the command line when a macro is running on the controller. The macro content and motion commands received from the command line can overwrite each other.
	Additional macro commands and information see "Controller Macros" (p. 64).	

#### INFORMATION

Absolute target positions can only be commanded if the axis was referenced beforehand; see "Referencing" (p. 34).

#### Motion in open-loop operation

Motion is triggered with the **SMO** command that directly specifies the control value for the PLine® driver electronics in the C-877.

The motion triggered by commands can be stopped using the following commands:

- #24, STP: abrupt stop
- HLT: gentle stop

In both cases, the error code 10 is set for information.

### 3.7.7 Generation of the Dynamics Profile

In closed-loop operation, the profile generator performs calculations to specify the target position, velocity, and acceleration of the axis for any point in time during motion (dynamics profile). The values calculated are called commanded values. The dynamics profile generated by the profile generator of the C-877 depends on the motion parameters that are given by commands (p. 82) and parameters:

Motion parameter	Commands	Parameter	Remarks
Acceleration (A)	ACC ACC?	Acceleration in closed-loop operation (parameter 0xB; physical unit of length/s <sup>2</sup> ); change with the ACC command or with SPA / SEP; can be saved with WPA.	Is limited by parameter 0x4A (maximum acceleration in closed-loop operation)
Deceleration (D)	DEC DEC?	Deceleration in closed-loop operation (parameter 0xC; physical unit of length/s <sup>2</sup> ); change with the DEC command or with SPA / SEP; can be saved with WPA.	Is limited by parameter 0x4B (maximum deceleration in closed-loop operation)
Velocity (V)	VEL VEL?	Velocity in closed-loop operation (parameter 0x49; physical unit of length/s); change with the VEL command or with SPA / SEP; can be saved with WPA.	Is limited by parameter 0xA (maximum velocity in closed-loop operation).
Target position at the end of the motion	MOV MVR GOH STE	-	The C-877 sets the target position to the current position of the axis in the following cases: <ul style="list-style-type: none"> <li>Switching on the servo mode with the SVO command</li> <li>Stopping the motion with the #24, STP, or HLT commands</li> </ul>

The profile generator of the C-877 only supports trapezoidal velocity profiles: The axis accelerates linearly (based on the acceleration value specified) until it reaches the specified velocity. It continues to move with this velocity until it decelerates linearly (based on the deceleration value specified) and stops at the specified target position.



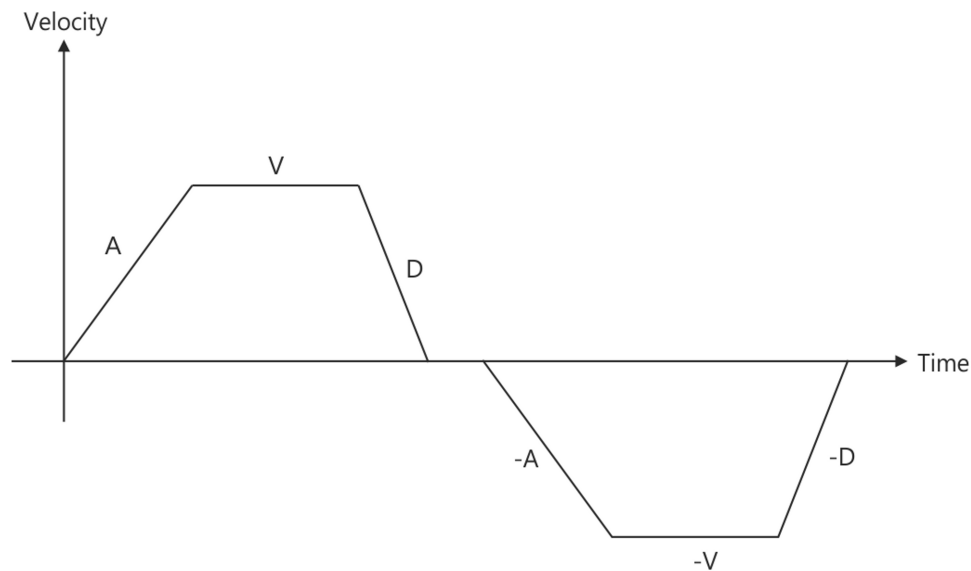


Figure 4: Basic trapezoidal velocity profile; A = acceleration, D = deceleration, V = velocity

If the deceleration has to begin before the axis reaches the specified velocity, the profile will not have a constant velocity portion and the trapezoid becomes a triangle.

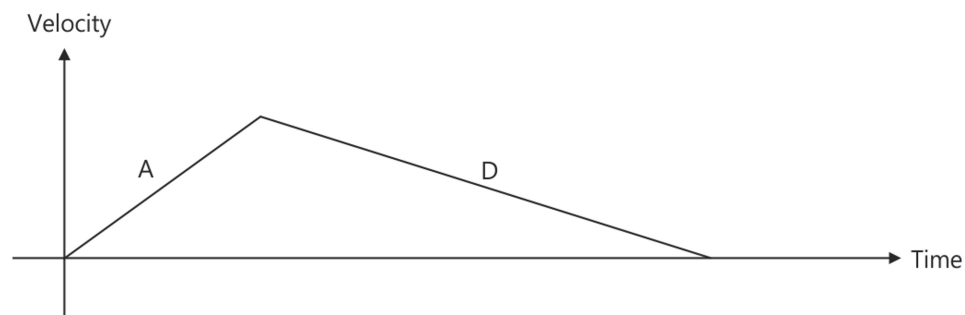


Figure 5: Basic trapezoidal velocity profile; A = acceleration, D = deceleration, no constant velocity

The edges for acceleration and deceleration can be symmetrical (acceleration = deceleration) or asymmetrical (acceleration  $\neq$  deceleration). The acceleration value is always used at the start of the motion. After that, the acceleration value is used during an increase in the absolute velocity and the deceleration value during a decrease in the absolute velocity. If no motion parameters are changed during the course of the motion, the acceleration value is used until the maximum velocity is reached and the deceleration value is used for the decrease in velocity down to zero.

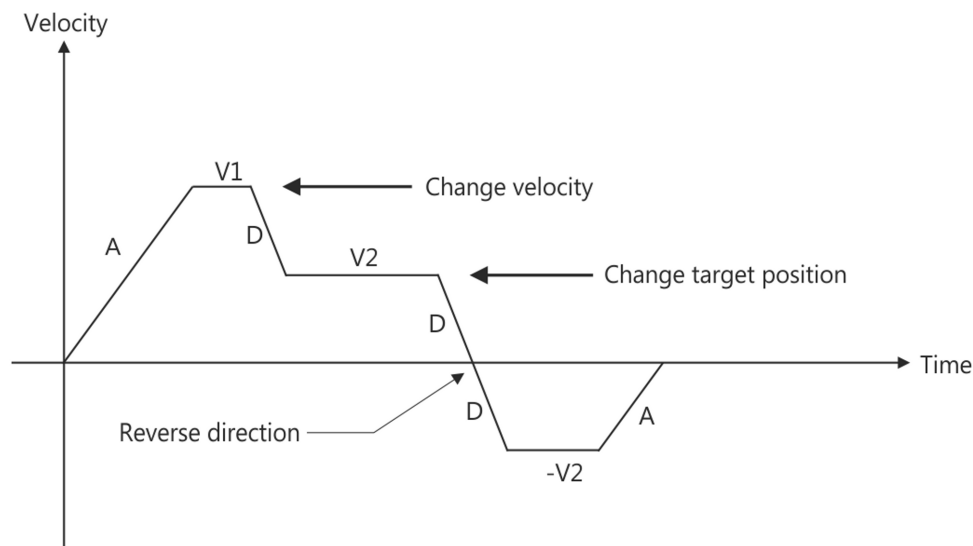


Figure 6: Complex trapezoidal profile with parameter changes; A = acceleration; D = deceleration; V1, V2, -V2 = velocities

All motion parameters can be changed while the axis is in motion. The profile generator will always attempt to stay within the permissible motion limits specified by the motion parameters. If the target position is changed during the motion so that overshooting is unavoidable, the profile generator will decelerate to a complete stop and reverse the direction of motion in order to reach the specified position.

### 3.7.8 Servo Algorithm and Other Control Value Corrections

In closed-loop operation, the following corrections optimize the control value for the PLine® driver electronics integrated in the C-877 and therefore the settling behavior of the system:

- **Servo algorithm:** The position error, which results from the difference between the calculated dynamics profile (see "Generation of Dynamics Profile" (p. 18)) and the actual position (sensor feedback), runs through a PID servo algorithm (proportional integral derivative).
- **Dynamics profile corrections:** The dynamics profile generated can be subject to an offset correction and a feed-forward control of the velocity.

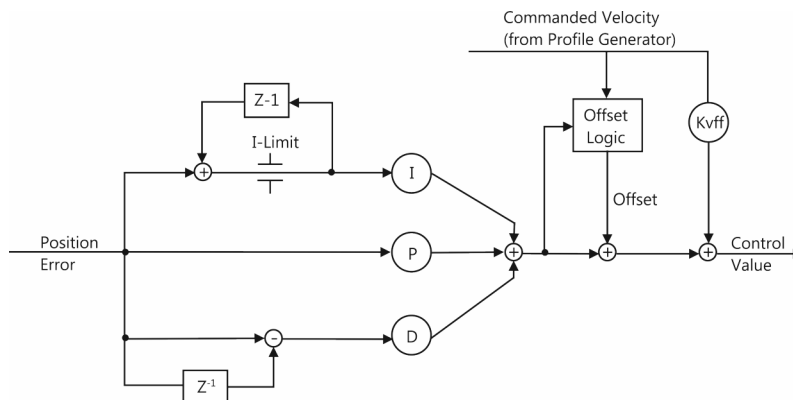


Figure 7: PID algorithm, offset compensation, and feed-forward control of the velocity (KVff)

For finer corrections, the C-877 switches between parameter groups 0 to 4 during the axis motion in closed-loop operation. The switching is done on the basis of configurable position windows.

Parameter groups 0 to 4 each contain the following settings:

- P, I, D terms and I limit for the servo algorithm
- Kvff term for the feed-forward control of the velocity
- Window limits for entry and exit

## INFORMATION

For compatibility reasons, the C-877 has an additional group of parameters for the servo algorithm and feed-forward control of the velocity:

- Parameters 0x1, 0x2, 0x3, 0x4, and 0x5

The values of these parameters are set automatically to the values of the servo control parameters of group 1 (0x411, 0x412, 0x413, 0x414, and 0x415).

- To optimize the dynamic behavior of the system, use the parameters 0x411, 0x412, 0x413, 0x414, and 0x415 (do **not** use: 0x1 to 0x5); see "Optimizing Servo Control Parameters" (p. 56).

## Servo algorithm

The servo algorithm uses the following servo control parameters. The optimum servo control parameter setting depends on your application and your requirements; see "Optimizing Servo Control Parameters" (p. 56).

Parameter	Description and Possible Values
<b>D Term Delay (No. Of Servo Cycles)</b> 0x71	D term delay The D term can be calculated as a floating average over several servo cycles. The parameter specifies how many values (i.e., servo cycles) are to be used for averaging.

Parameter	Description and Possible Values
<b>P term 0</b> 0x401 <b>P term 1</b> 0x411 <b>P term 2</b> 0x421 <b>P term 2</b> 0x431 <b>P term 2</b> 0x441	Proportional constants (dimensionless) of parameter groups 0 to 4 0 to 65535 Aim: Rapid correction of the position error
<b>I term 0</b> 0x402 <b>I term 1</b> 0x412 <b>I term 2</b> 0x422 <b>I term 2</b> 0x432 <b>I term 2</b> 0x442	Integral constants (dimensionless) of parameter groups 0 to 4 0 to 65535 Aim: Reduction of the static position error
<b>D term 0</b> 0x403 <b>D term 1</b> 0x413 <b>D term 2</b> 0x423 <b>D term 2</b> 0x433 <b>D term 2</b> 0x443	Differential constants (dimensionless) of parameter groups 0 to 4 0 to 65535 Aim: Damping of rapid control oscillation
<b>I limit 0</b> 0x404 <b>I limit 1</b> 0x414 <b>I limit 2</b> 0x424 <b>I limit 2</b> 0x434 <b>I limit 2</b> 0x444	Limitation of the integral constants (dimensionless) of parameter groups 0 to 4 0 to 65535

### INFORMATION

To prevent a servo jitter of the axis after the target position is reached, the I term of the parameter group used for step-and-settle (group 0 by default) should be deactivated or minimized.

- Use the corresponding I limit to deactivate or minimize an I term. Example: I term 0 (0x402) is deactivated when I limit 0 (0x404) has the value zero (default setting)

The input of the servo algorithm can be configured for the C-877 with the following parameters:

Parameter	Description and Possible Values
<b>Numerator Of The Servo-Loop Input Factor</b> 0x5A	Numerator and denominator of the servo-loop input factor 1 to 1,000,000 for both parameters The servo-loop input factor decouples the servo control parameters from the encoder resolution.
<b>Denominator Of The Servo-Loop Input Factor</b> 0x5B	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.

### Corrections of the dynamics profile

The dynamics profile corrections for closed-loop operation can be configured via the parameters listed below:

Parameter	Description and Possible Values
<b>Motor Offset Positive</b> 0x33	Offset for the positive direction of motion (dimensionless). 0 to 32767 The range of values corresponds to 0 to 10 V control voltage. Compensates the internal preload of the piezomotor.
<b>Motor Offset Negative</b> 0x34	Offset for the negative direction of motion (dimensionless). 0 to 32767 The range of values corresponds to 0 to -10 V control voltage. Compensates the internal preload of the piezomotor.
<b>Motor Drive Offset</b> 0x48	Velocity-dependent offset (dimensionless). Is used if the commanded velocity does not equal zero (i.e., if the end of the dynamics profile has not been reached yet). 0 to 32767 The range of values corresponds to 0 to 10 V control voltage. Depending on the current direction of motion, the offset value has a positive or negative sign.
<b>Kvff 0</b> 0x405 <b>Kvff 1</b> 0x415 <b>Kvff 2</b> 0x425 <b>Kvff 2</b> 0x435 <b>Kvff 2</b> 0x445	Feed-forward control of the commanded velocity for parameter groups 0 to 4 0 to 65535 Aim: Minimization of the position error

### INFORMATION

To start motion, PLine® ultrasonic piezo motors require a particular piezo voltage that is not equal to zero. For this reason, offset values (parameters 0x33, 0x34, 0x48) are added to the control value and therefore to the control voltage. The offset values for the positive and negative direction of motion (0x33 and 0x34) are to be kept as low as possible with a velocity-dependent offset (0x48). The optimum offset values for the positive and negative direction of motion can strongly deviate from each other especially in the case of a vertically aligned motion axis.

### Switching between parameter groups 0 to 4

Switching between parameter groups 0 to 4 for servo algorithm and feed-forward control of the velocity can be configured with the parameters listed in the following.

Parameter	Description and Possible Values
<b>Servo Window Mode</b> 0x4D	Reference variable for the position windows 0 = Target position (default setting) 1 = Commanded position (from dynamics profile) This parameter specifies the reference variable for the position

Parameter	Description and Possible Values
	windows that are used to switch between parameter groups 0 to 4 for servo algorithm and feed-forward control. The switching is done based on the difference between the current position and the selected reference variable.
<b>Number Of Servo Parameter Groups</b> 0x400	Maximum number of parameter groups used 1 to 5 These parameters indicate the maximum number of parameter groups among which are switched during the axis motion. Default setting: 3 (parameter groups 0 to 2 are used)
<b>Window Enter 0</b> 0x406 <b>Window Enter 1</b> 0x416 <b>Window Enter 2</b> 0x426 <b>Window Enter 3</b> 0x436 <b>Window Enter 4</b> 0x446	Position windows for activating parameter groups 0 to 4 0 to $2^{31}$ counts of the encoder The parameters specify the entrance windows for the parameter groups. The windows are centered around the reference variable selected with the parameter 0x4D. When the current position enters the entrance window of a parameter group, this parameter group is activated. The parameter values each correspond to half of the window width. They can only be changed if the servo mode is switched off.
<b>Window Exit 0</b> 0x407 <b>Window Exit 1</b> 0x417 <b>Window Exit 2</b> 0x427 <b>Window Exit 3</b> 0x437 <b>Window Exit 4</b> 0x447	Position window for deactivating parameter groups 0 to 4 0 to $2^{31}$ counts of the encoder The parameters specify the exit windows for the parameter groups. The windows are centered around the reference variable selected with the parameter 0x4D. When the current position leaves the exit window of a parameter group, this parameter group is deactivated, and the next-highest parameter group is activated. The parameter values each correspond to half of the window width. They can only be changed if the servo mode is switched off.

### INFORMATION

The following applies to the position windows:

- The entrance window for parameter group n must be smaller than the entrance window for parameter group n+1.
- The exit window for parameter group n must be smaller than the exit window for parameter group n+1.
- The position windows of the "outermost" parameter group used are ignored. Which parameter group is the outermost group used depends on the setting of the parameter **Number Of Servo Parameter Groups** (0x400). Example: Parameter 0x400 has the value 3. The switching is then done between parameter groups 0, 1, and 2. Parameter group 2 is the outermost parameter group used. Because the position windows of parameter group 2 are ignored, it remains activated even when the current position is outside of its exit window (0x427).
- The entrance and exit windows for parameter group 0 are also used as settling windows

for determining the on-target state (p. 27).

- For a stable switching behavior, the exit window of a parameter group should be larger than its entrance window.

The following two figures show the switching between parameter groups during axis motion. Settings in the examples:

- Reference variable of the switching: Target position (upper figure) or commanded position (lower figure)
- The entrance windows of the parameter groups are smaller than their exit windows.
- The maximum number of parameter groups used is 3.
- When the current position enters the entrance window for parameter group 0, parameter group 0 is activated.

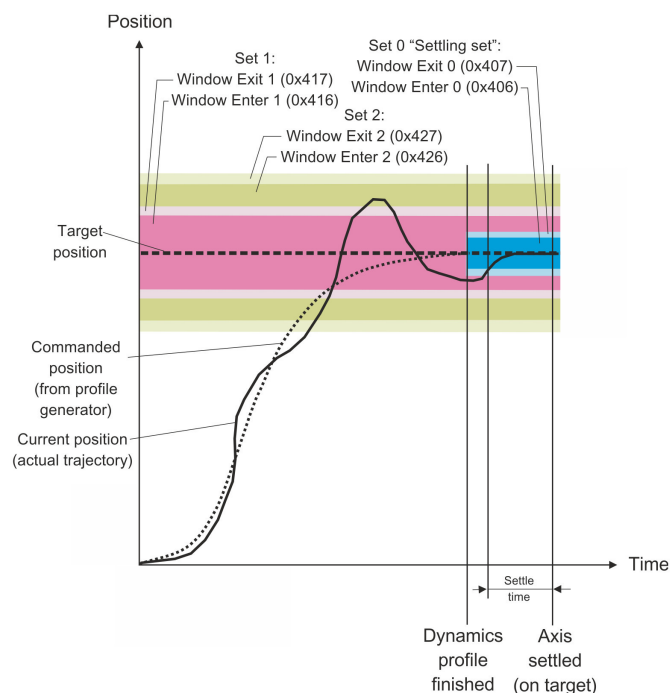


Figure 8: Switching between parameter groups 0 to 2 based on the difference between the current position and the target position

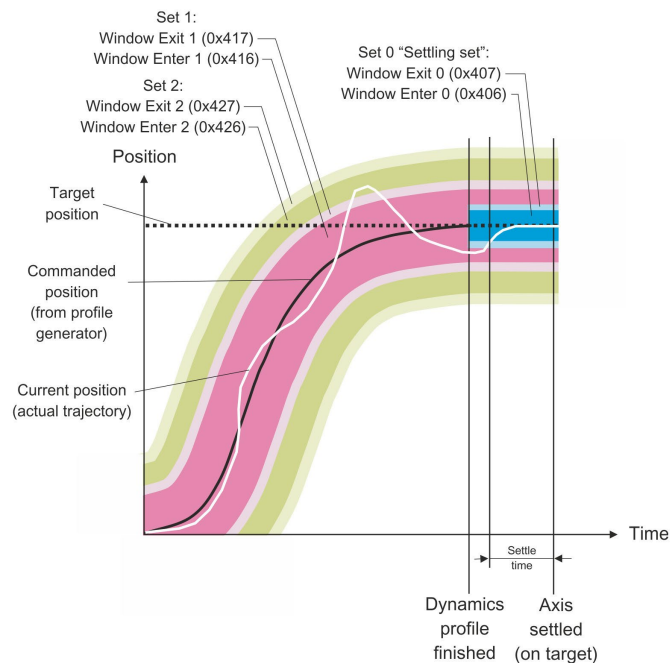


Figure 9: Switching between parameter groups 0 to 2 based on the difference between the current position and the commanded position

### 3.7.9 Optional Two-Phase Control

The two-phase control (p. 2) is enabled by default. Depending on the application, the two-phase control can be enabled to improve the settling behavior. To enable the two-phase control, the switch-on/pause ratio for parallel driving of both piezo segments of a PLine® piezomotor must be configured accordingly using the parameters listed below (groups 0 to 4).

The switching between parameter groups 0 to 4 takes place during the axis motion in closed-loop operation. To switch between the parameter groups, the C-877 uses the parameters that are also used for the servo algorithm, see "Servo Algorithm and Further Control Value Corrections" (p. 20).

Parameter	Description and Possible Values
<b>2nd Phase On 0 (No. Of Servo Cycles)</b> 0x409	Duty cycle of parallel control of both piezo segments; for parameter groups 0 to 4 0 (default setting) up to 34464 servo cycles
<b>2nd Phase On 1 (No. Of Servo Cycles)</b> 0x419	
<b>2nd Phase On 2 (No. Of Servo Cycles)</b> 0x429	
<b>2nd Phase On 3 (No. Of Servo Cycles)</b> 0x439	
<b>2nd Phase On 4 (No. Of Servo Cycles)</b> 0x449	



Parameter	Description and Possible Values
<b>2nd Phase Off 0 (No. Of Servo Cycles)</b> 0x40A	Pause duration of the parallel control of both piezo segments; for parameter groups 0 to 4 0 (default setting) up to 34464 servo cycles
<b>2nd Phase Off 1 (No. Of Servo Cycles)</b> 0x41A	
<b>2nd Phase Off 2 (No. Of Servo Cycles)</b> 0x42A	
<b>2nd Phase Off 3 (No. Of Servo Cycles)</b> 0x43A	
<b>2nd Phase Off 4 (No. Of Servo Cycles)</b> 0x44A	

### 3.7.10 On-Target State

In closed-loop operation, the on-target state can be used to check whether the target position has been reached:

- On-target state = true (1): the target position is considered as reached
- On-target state = false (0): the target position is considered as not reached

The C-877 determines the on-target state on the basis of the following criteria:

- Settling window around the target position, is given by the entrance and exit windows for parameter group 0 (parameter 0x406 and 0x407)
- Delay time for setting the on-target state (parameter 0x3F)

The on-target state has the value **true** in the following cases:

- The current position is inside the settling window and stays there at least for the duration of the delay time.
- If the value for the delay time is set to 0: The end of the dynamics profile is reached.

The on-target state can be read with the `ONT?`, `#4` and `SRG?` commands.

Parameter	Description and Possible Values
<b>Settling Time (s)</b> 0x3F	Delay time for setting the on-target state 0 to 1.000 s
<b>Window Enter 0 (encoder counts)</b> 0x406	Settling window around the target position 0 to $2^{31}$ counts of the encoder The parameters give the window limits for entry and exit. If the current position exits the settling window, the target position is no longer considered as reached. The parameter values each correspond to half of the window width. They can only be changed if the servo mode is switched off.
<b>Window Exit 0 (encoder counts)</b> 0x407	

### 3.7.11 Supported Motor Types

The C-877 supports all types of PLine® positioners and drives integrated in PLine® ultrasonic piezomotors currently offered by PI. The adaptation to the connected motor type is done using the following parameters:

Parameter	Description and Possible Values
<b>Maximum Motor Output (V)</b> 0x7C	Maximum permissible piezo voltage 0 to 71 V <sub>rms</sub> This parameter determines the maximum permissible amplitude of the output piezo voltage.
<b>Output Frequency (kHz)</b> 0x51	Frequency of the piezo voltage 0 to 500 kHz This parameter determines the frequency with which the output piezo voltage oscillates in order to excite the piezo actuator in the PLine® ultrasonic piezomotor. Sources for the value of the parameter: <ul style="list-style-type: none"> <li>When the frequency control (p. 28) is switched on and active: Determination by the frequency control</li> <li>Direct modification, e.g., with the SPA command (simultaneously switches off the frequency control)</li> </ul>

#### INFORMATION

Further information can be found in the user manual of your PLine® positioner.

### 3.7.12 Automatic Frequency Control

The C-877 is equipped with a frequency control that optimizes the frequency of the output piezo voltage. At the optimum operating point, the frequency of the piezo voltage is as close as possible to the resonant frequency of the connected motor. The resonant frequency of the motor is influenced by various factors:

- Motor type
- Installation conditions of the motor
- Execution of the run-in procedure
- Temperature

The frequency control operates with 1 kHz.

The frequency control can be configured via the parameters listed below:

Parameter	Description and Possible Values
<b>Frequency Control</b> 0x52	State of the frequency control 0 = Frequency control switched off 1 = Frequency control switched on (default setting) When the frequency control is switched on <b>and</b> active, it sets the

Parameter	Description and Possible Values
	value of the <b>Output Frequency (kHz)</b> parameter (0x51, see "Supported Motor Types" (p. 28)). The criterion for activating the frequency control is given by parameter 0x55. Direct modification of the parameter 0x51 (e.g., with the SPA command) simultaneously switches off the frequency control.
<b>Minimum Output Frequency (kHz)</b> 0x53	Minimum frequency of the piezo voltage (kHz) 0 to 500 kHz This parameter gives the smallest possible value for parameter 0x51 when the frequency control is switched on and active.
<b>Maximum Output Frequency (kHz)</b> 0x54	Maximum frequency of the piezo voltage (kHz) 0 to 500 kHz This parameter gives the largest possible value for parameter 0x51 when the frequency control is switched on and active.
<b>Minimum Motor Output For Frequency Control</b> 0x55	Minimum control value for activating the frequency control 0 to 32767 When the modulus of the current control value is at least as large as the value of this parameter, the switched-on frequency control becomes active and sets the value of the parameter 0x51.

### 3.7.13 Reference Switch Detection

The C-877 receives the signal from a reference switch on pin 13 of the **Motor** socket (p. 195).

The following parameters can be used to configure how the C-877 detects the reference switch:

Parameter	Description and Possible Values
<b>Invert Reference?</b> 0x31	Should the reference signal be inverted? 0 = Reference signal not inverted 1 = Reference signal inverted This parameter is used for inverting the reference signal whose source can be either the reference switch or a digital input which is used instead of the reference switch .
<b>Has Reference?</b> 0x14	Does the positioner have a reference switch? 0 = No reference switch installed 1 = Reference switch installed (signal input at <b>Motor</b> socket) This parameter enables or disables reference moves to the reference switch installed.
<b>Reference Signal Type</b> 0x70	Reference signal type 0 = Direction-sensing reference switch (default setting). 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 approach takes place via the negative limit switch. 3 = Index pulse. The approach takes place via the positive limit switch.

The signal from the reference switch of the positioner can be used for reference moves. In the case of a positioner with incremental position sensor, the controller knows the absolute axis position after a reference move to the reference switch; see "Referencing" (p. 34).

### 3.7.14 Limit Switch Detection

The C-877 receives limit switch signals at the **Motor** socket (p. 195):

- Pin 5: Positive limit switch
- Pin 12: Negative limit switch

The following parameters can be used to configure how the C-877 detects the limit switches:

Parameter	Description and Possible Values
<b>Limit Mode</b> 0x18	Signal logic of the limit switches 0 = Positive limit switch active high (pos-HI), negative limit switch active high (neg-HI) 1 = Positive limit switch active low (pos-LO), neg-HI 2 = pos-HI, neg-LO 3 = pos-LO, neg-LO
<b>Has No Limit Switches?</b> 0x32	Does the positioner have limit switches? 0 = Positioner has limit switches (signal inputs at the <b>Motor</b> socket) 1 = Positioner has no limit switches This parameter activates or deactivates motion stopping at the limit switches installed.
<b>Use Limit Switches Only For Reference Moves?</b> 0x77	Should the limit switches only be used for reference moves? 0 = Use limit switches for stopping at the end of the travel range and for reference moves (default) 1 = Use limit switches only for reference moves This parameter is intended for use with rotation stages. This parameter is only evaluated when the parameter 0x32 has the value 0.

The signals from the limit switches (also end-of-travel sensors) of a linear positioner are used to stop the motion prior to the hard stop at both ends of the travel range. Because the set deceleration is not taken into account here, there is a risk at high velocities that the positioner will strike the hard stop anyway. To prevent this, soft limits (p. 30) can be set via parameters of the C-877.

The limit switch signals can also be used for reference moves. In the case of a positioner with incremental position sensor, the controller knows the absolute axis position after a reference move to a limit switch; see "Referencing" (p. 34).

### 3.7.15 Travel Range and Soft Limits

The following parameters of the C-877 reflect the physical travel range of the positioner and define soft limits:

Parameters	Description and Possible Values
<b>Maximum Travel In Positive Direction (Phys. Unit)</b> 0x15	Soft limit in positive direction (physical unit) Based on the zero position. If this value is smaller than the position value for the positive limit switch (which results from the sum of the parameters 0x16 and 0x2F), the positive limit switch cannot be used for reference moves. The value can be negative.
<b>Value At Reference Position (Phys. Unit)</b> 0x16	Position value at the reference switch (physical unit) The current position is set to this value if the axis has performed a reference move to the reference switch. The parameter value is used in addition for calculating the position values which are set after reference moves to the limit switches; this also applies when the mechanics does not have a reference switch.
<b>Distance From Negative Limit To Reference Position (Phys. Unit)</b> 0x17	Distance between reference switch and negative limit switch (physical unit) If the axis has performed a reference move to the negative limit switch, the current position is set to the difference between the values of parameters 0x16 and 0x17.
<b>Distance From Reference Position To Positive Limit (Phys. Unit)</b> 0x2F	Distance between reference switch and positive limit switch (physical unit) If the axis has performed a reference move to the positive limit switch, the current position is set to the sum of the values of parameters 0x16 and 0x2F.
<b>Maximum Travel In Negative Direction (Phys. Unit)</b> 0x30	Soft limit in negative direction (physical unit) Based on the zero position. If this value is larger than the position value for the negative limit switch (which results from the difference between the parameters 0x16 and 0x17), the negative limit switch cannot be used for reference moves. The value can be negative.
<b>Range Limit Min</b> 0x07000000	Additional soft limit for the negative direction of motion (physical unit) If the current position reaches this value in either closed-loop or open-loop operation, the control value is set to zero and the motion is stopped as a result. The axis can move again as soon as the value for the soft limit has been decreased.
<b>Range Limit Max</b> 0x07000001	Additional soft limit for the positive direction of motion (physical unit) If the current position reaches this value in either closed-loop or open-loop operation, the control value is set to zero and the motion is stopped as a result. The axis can move again as soon as the value for the soft limit has been increased.

**INFORMATION**

The C-877 supports two parameter pairs for establishing soft limits. They are intended for different applications:

- 0x15 (**Maximum Travel In Positive Direction (Phys. Unit)**) and 0x30 (**Maximum Travel In Negative Direction (Phys. Unit)**):
  - The limits establish the permissible travel range in closed-loop operation.
  - Motion commands are executed only if the commanded position is within these soft limits.
  - The limits always refer to the current zero position.
  - Appropriate values are loaded when the positioner type is selected from the positioner database.
- 0x07000000 (**Range Limit Min**) and 0x07000001 (**Range Limit Max**):
  - Using these limits is recommended only if open-loop motion is required. For logical reasons, the values are outside the soft limits which are specified via 0x15 and 0x30.
  - Apply both in closed-loop and open-loop operation.
  - Motions are stopped abruptly once the current position reaches a limit.
  - The limits are independent of the current zero position.
  - The values are not loaded from the positioner database and are set in the default settings so that the limits are deactivated.

**Examples**

The following examples refer to an axis of a positioner with incremental sensor, reference switch and limit switches.

The distance between the negative and positive limit switches of the axis is 20 mm. The reference switch has a distance of 8 mm to the negative limit switch and a distance of 12 mm to the positive limit switch.

This switch setup of the axis is reflected in the following parameters:

- Parameter 0x17: Distance between negative limit switch and reference switch = 8 mm
- Parameter 0x2F: Distance between reference switch and positive limit switch = 12 mm

**INFORMATION**

The switch setup of the axis can be determined with the `FED` and `POS?` commands.

**Example 1: Maximum travel range available**

After reference moves (p. 34), the current position is to have the following values:

- Move to the negative limit switch (start with `FNL`): current position = 0
- Move to the reference switch (start with `FRF`): current position = 8
- Move to the positive limit switch (start with `FPL`): current position = 20

As a result, parameter 0x16, which specifies the position value for the reference switch and is included in the calculation of the position values for the limit switches during reference moves, has the value 8.

The travel range is not to be limited by soft limits. As a result, the respective parameters are set as follows:

- Parameter 0x15 = 20
- Parameter 0x30 = 0

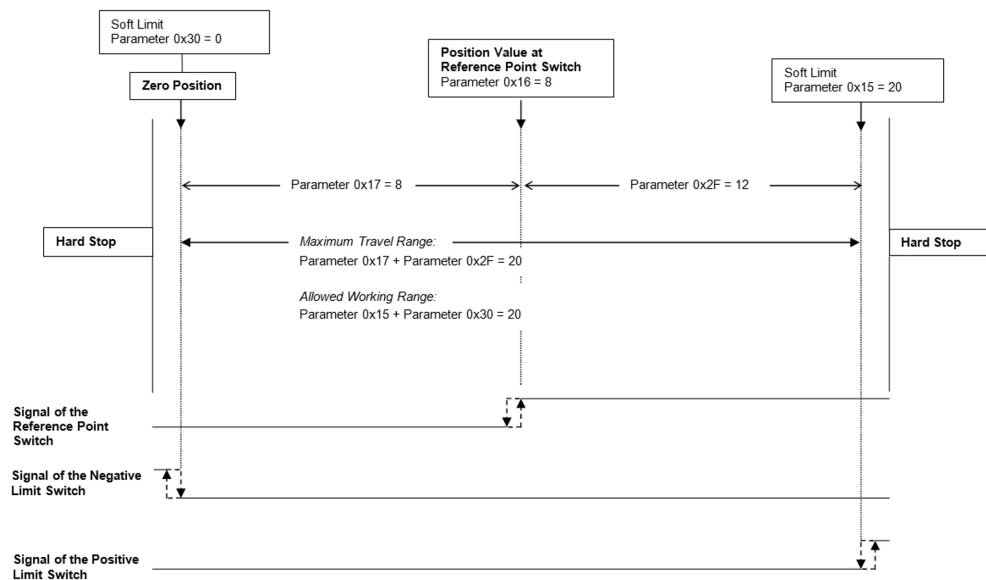


Figure 10: The travel range of the axis is not limited by soft limits.

After a reference move of the axis to the reference switch (**FRF** command), query commands return the following responses:

- **TMN?** returns the value 0
- **TMX?** returns the value 20
- **POS?** returns the value 8

#### Example 2: Travel range limited by soft limits

The zero position should be located at approximately a third of the distance between the negative limit switch and the reference switch. As a result, parameter 0x16 now has the value 5.4.

A safety distance is to be put in place at both ends of the travel range by establishing soft limits. As a result, the soft limits are set as follows:

- Parameter 0x15 = 16.4
- Parameter 0x30 = -2.1

According to that, the axis can move 16.4 mm from the zero position in the positive direction and 2.1 mm in the negative direction respectively. The limit switches can no longer be used for reference moves.

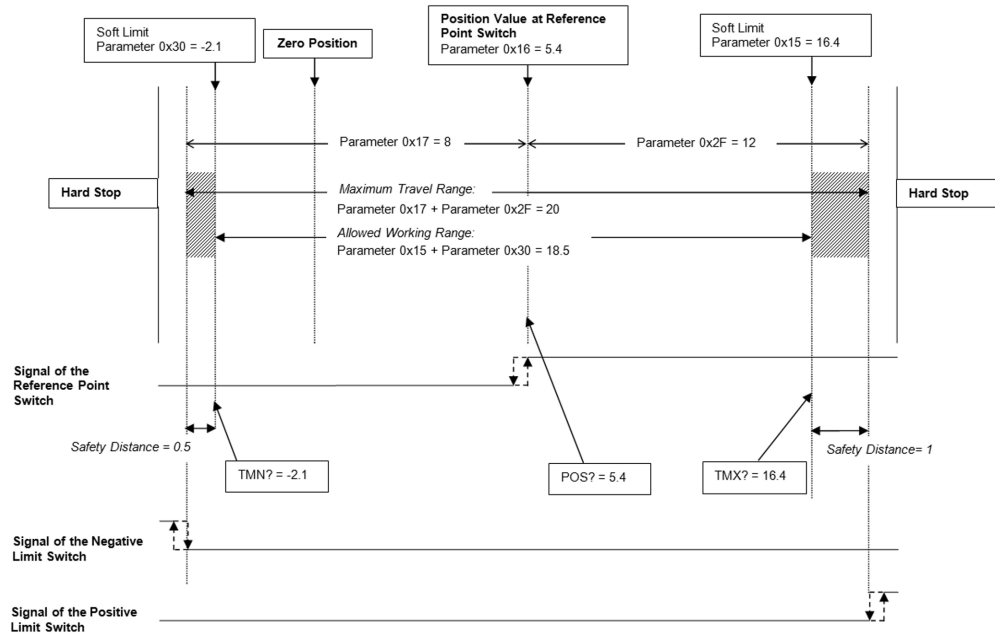


Figure 11: The travel range of the axis is limited by soft limits.

After a reference move of the axis to the reference switch (**FRF** command), query commands return the following responses:

- **TMN?** returns the value -2.1
- **TMX?** returns the value 16.4
- **POS?** returns the value 5.4

### 3.7.16 Referencing

The C-877 only supports positioners that are equipped with an incremental position sensor. Incremental sensors only supply relative motion information. Therefore, the controller does not know the absolute position of the axis when it is switched on or rebooted. Before absolute target positions can be commanded and reached, the axis must be referenced beforehand.

Referencing can be done in different ways:

- **Reference move (default)**: A reference move moves the axis to a defined point, e.g., to the reference switch or to a limit switch. At this point, the current position is set to a defined value. The controller now knows the absolute axis position.
- **Setting the absolute position manually**: If this referencing method was activated by the **RON** command (p. 118), you can set the current position of the axis to an arbitrary value at an arbitrary point using the **POS** command (p. 116). The axis is not moved here. The controller knows the absolute axis position afterwards.



**INFORMATION**

During startup using PIMikroMove, referencing is done via a reference move by default. Knowledge of the commands and parameters described here is not needed for referencing using PIMikroMove.

**INFORMATION**

To achieve maximum repeatability when referencing, each reference move comprises the following steps:

1. First move to the switch selected. The maximum velocity is specified via parameter 0x49 (**Closed-Loop Velocity (Phys. Unit/s)**, equivalent to setting with the VEL command).
2. Stop on reaching the switch edge. The higher the velocity on approach, the farther the axis overruns the edge of the switch (overshooting).
3. Move in the opposite direction to compensate for overshoot.
4. Second move to the switch selected. The maximum velocity is specified via parameter 0x50 (**Velocity For Reference Moves (Phys. Unit/s)**, specific velocity for reference moves only).
5. Stop when reaching the switch edge.
6. Move in the opposite direction to compensate for overshoot.
7. Set the current position to a defined value, referencing is finished.

The lower the velocity is when approaching the switch, the less the overshoot will be and the higher the repeatability. Therefore, the maximum value of parameter 0x50 should be as large as the value of parameter 0x49, though ideally substantially less.

The actual velocities during the reference move are calculated from the values of the following parameters and can be lower than the maximum values.

- Parameter 0x49 or 0x50
- Parameter 0x63 (**Distance Between Limit And Hard Stop (Phys. Unit)**)
- Parameter 0xC (**Closed-Loop Deceleration (Phys. Unit/s<sup>2</sup>)**)

**Commands**

The following commands are available for referencing:

Com-mand	Syntax	Function
RON	RON {<AxisID> <ReferenceOn>}	<p>Sets the referencing method (&lt;ReferenceOn&gt;) for the axis:</p> <ul style="list-style-type: none"> <li>▪ 0: An absolute position value can be assigned with POS or TSP to reference the axis, or a reference move can be started with FRF, FNL or FPL.</li> <li>▪ 1 (default): A reference move must be started with FRF, FNL or FPL to reference the axis. Using POS is not permitted.</li> </ul>

Com-mand	Syntax	Function
RON?	RON? [{<AxisID>}]	Gets referencing method.
FRF	FRF [{<AxisID>}]	Starts a reference move to the reference switch. The approach depends on the value of the <b>Reference Signal Type</b> parameter (ID 0x70): <ul style="list-style-type: none"> <li>0 or 1: The approach always takes place from the same side irrespective of the axis position when the command is sent.</li> <li>2: The approach takes place via the negative limit switch.</li> <li>3: The approach takes place via the positive limit switch.</li> </ul>
FRF?	FRF? [{<AxisID>}]	Queries whether the reference point for an axis has already been defined. 1 = Reference point has been defined 0 = Reference point has not been defined
FNL	FNL [{<AxisID>}]	Starts a reference move to the negative limit switch.
FPL	FPL [{<AxisID>}]	Starts a reference move to the positive limit switch.
POS	POS {<AxisID> <Position>}	Sets the current position (does not trigger a motion) and therefore defines the reference point.

### Parameters

Reference moves can be configured with the following parameters:

Parameter	Description and Possible Values
<b>Closed-Loop Deceleration (Phys. Unit/s<sup>2</sup>)</b> 0xC	Deceleration in closed-loop operation For details, see "Generation of Dynamics Profile" (p. 18).
<b>Reference Travel Direction</b> 0x47	Default direction for the reference move 0 = automatic detection 1 = negative direction 2 = positive direction
<b>Closed-Loop Velocity (Phys. Unit/s)</b> 0x49	Velocity in closed-loop operation For details, see "Generating a Dynamics Profile" (p. 18).
<b>Velocity For Reference Moves (Phys. Unit/s)</b> 0x50	Velocity for reference move Specifies the maximum velocity during a reference move for the second approach of the switch selected. For high repeatability

Parameter	Description and Possible Values
	during referencing, the maximum of this value should as large as the value of parameter. If the value of parameter 0x50 is set to 0, reference moves are not possible.
<b>Distance Between Limit And Hard Stop (Phys. Unit)</b> 0x63	Distance between internal limit switch and hard stop Determines the maximum stopping distance during reference moves. The actual velocities during a reference move are calculated on the basis of this value, the deceleration set (0xC) and the velocities set (0x49 and 0x50).
<b>Distance From Limit To Start Of Ref Search (Phys. Unit)</b> 0x78	Distance between limit switch and the starting position for the motion to the index pulse. For details, see explanation below the table.
<b>Distance For Reference Search (Phys. Unit)</b> 0x79	Maximum distance for the motion to the index pulse For details, see explanation below the table.

The parameters 0x78 and 0x79 are used for reference moves when the two following conditions are met:

- The reference move is started with FRF.
- The **Reference Signal Type** parameter (0x70) has the value 2 or 3.

Sequence of the reference move:

1. The axis moves to the corresponding limit switch.
2. The axis moves the distance given by the parameter 0x78 away from the limit switch.
3. The axis moves to the index pulse and travels up to the maximum distance specified by parameter 0x79.

### INFORMATION

- For maximum repeatability, the reference move must always be done in the same way.

### INFORMATION

The limit switches can be used for reference moves only if the travel range is not limited by soft limits (p. 30).

### INFORMATION

If the absolute position of the axis is defined manually with the **POS** command, conflicts with the settings for the soft limits can occur (parameter 0x15, query with **TMX?**, and 0x30, query with **TMN?**).

- Set the absolute position of the axis manually only if referencing is not otherwise possible.

#### ***INFORMATION***

If the current parameter settings of the C-877 are written to the nonvolatile memory in PIMikroMove or by entering the WPA command using the password 100 or 101, the axis will no longer be considered "referenced" (the response to FRF? is 0).

---

## **4 Unpacking**

1. Unpack the C-877 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. 191).
4. Keep all packaging materials in case the product needs to be returned.



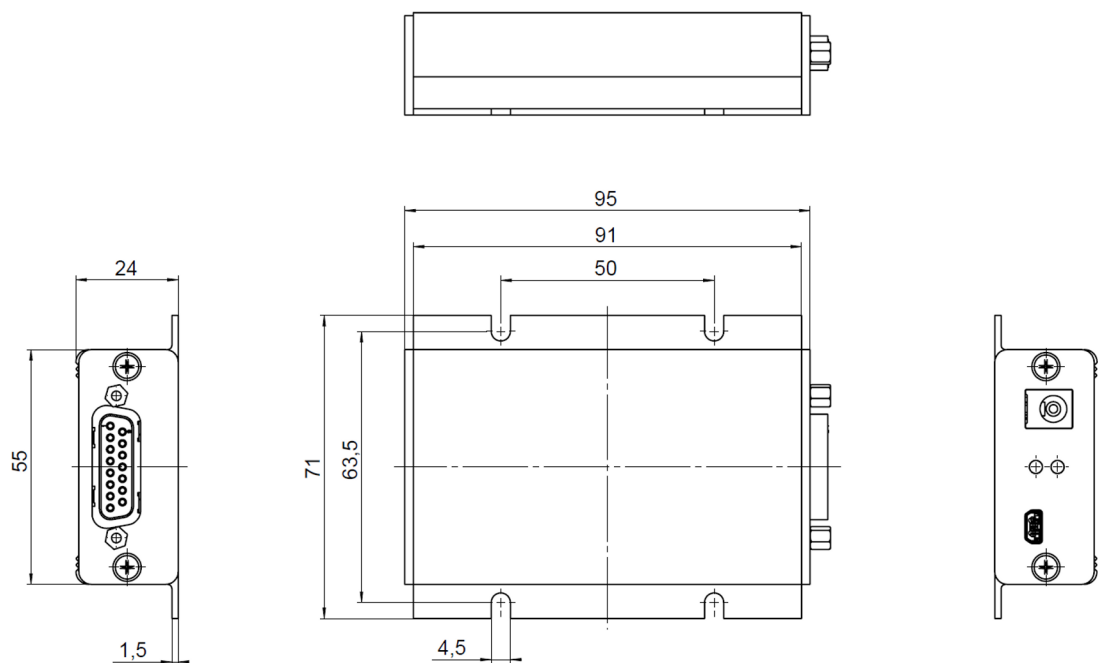
## 5 Installing

### 5.1 General Notes on Installation

- Install the C-877 near the power source so that the power plug can be quickly and easily disconnected from the mains.
- Only use cables and connectors that meet local safety regulations.

### 5.2 Mounting the C-877

The C-877 can be used as benchtop device or mounted in any orientation on a surface.



#### Tools and accessories

- Suitable screws
- Suitable screwdriver

#### Mounting the C-877

1. Make the necessary holes in the surface.

The arrangement of the recesses in the mounting rails of the C-877 can be found in the figure.

2. Use two screws on each side to fix the C-877 to the recesses in the mounting rails.

## 5.3 Connecting the C-877 to the Protective Earth Conductor

### INFORMATION

- Pay attention to the applicable standards for connecting the protective earth conductor.


#### Requirements

- ✓ You have read and understood the General Notes on Installation (p. 41).
- ✓ The C-877 is not connected to a power supply.

#### Tools and accessories

- Suitable protective earth conductor:
  - Cable cross section  $\geq 0.75 \text{ mm}^2$
  - Contact resistance  $< 0.1 \text{ ohm}$  at 25 A at all connection points relevant for mounting the protective earth conductor
- Fastening material for the protective earth conductor, sits on the protective earth connection (threaded bolt) in the following order on delivery of the C-877, starting with the housing:
  - Toothed washer
  - Nut
  - Flat washer
  - Safety washer
  - Nut
- Suitable wrench

#### Connecting the C-877 to the protective earth conductor

1. If necessary, attach a suitable cable lug to the protective earth conductor.
2. Remove the outer nut from the protective earth connection C-877 (threaded bolt (p. 7) marked with ).
3. Connect the protective earth conductor:
  - a) Push the cable lug of the protective earth conductor onto the threaded bolt.
  - b) Screw the nut onto the threaded bolt. In this way, the cable lug of the protective earth conductor is wedged between the toothed washer and the nut.
  - c) Tighten the nut with at least three turns and a torque of 1.2 Nm to 1.5 Nm.



## 5.4 Connecting the Power Adapter to the C-877

### Requirements

- ✓ The power cord is **not** connected to the power socket.

### Tools and accessories

- 24-V wide-range-input power supply included (for line voltages between 100 and 240 V alternating current at 50 or 60 Hz)  
Alternative: Suitable power supply that supplies 24 V direct current and a maximum output current of at least 0.8 amperes
- Power cord included  
Alternative: Suitable power cord

### Connecting the power supply to the C-877

- Connect the barrel connector of the power supply to the 24 V connection (**24 V** — — — **0.8 A**) of the C-877.
- Connect the power cord to the power supply.

## 5.5 Connecting the Positioner

### NOTICE



#### Damage if a wrong motor is connected!

Connecting a positioner with DC motor, stepper motor, or voice coil drive to the C-877 can cause irreparable damage to the positioner or controller.

- Only connect a positioner with PLine® ultrasonic piezo motors to the C-877.

### Requirements

- ✓ The C-877 is switched off, i.e., the power supply is **not** connected to the power socket with the power cord.
- ✓ You have read and understood the user manual of the positioner.

### Connecting the positioner

1. Connect the positioner to the **Motor** socket of the C-877.
2. Use the integrated screws to secure the connections against accidental disconnection.

## 5.6 Installing the PC Software

Communication between the C-877 and a PC is required to configure the C-877 and to command motion using the GCS commands. Various PC software applications are available for this purpose.

### 5.6.1 Doing Initial Installation

#### Accessories

- PC with Windows or Linux operating system and at least 30 MB free storage space
  - Data storage device with PI Software Suite (included in the scope of delivery)
- For information on the compatibility of the software with PC operating systems see the C-990.CD1 Release News in the root directory of the data storage device.

#### Installing the PC software on Windows

1. Start the installation wizard by double-clicking **PISoftwareSuite.exe** in the installation directory (root directory of the data storage device).

The **InstallShield Wizard** window opens for installing the PI Software Suite.

2. Follow the instructions on the screen.

The PI Software Suite includes the following components:

- Drivers for use with NI LabVIEW software
- Dynamic program library for GCS
- PIMikroMove
- PC software for updating the firmware of the C-877
- PIUpdateFinder for updating the PI Software Suite
- USB driver

#### Installing the PC software on Linux

1. Unpack the tar archive from the /Linux directory of the data storage device to a directory on your PC.
2. Open a terminal and go to the directory to which you have unpacked the tar archive.
3. Log in as a superuser (root privileges).
4. To start the installation, enter `./INSTALL`  
Pay attention to capitalization while entering the command.
5. Follow the instructions on the screen.

You can select individual components for installation.

## 5.6.2 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 PIUpdateFinder manual (A000T0028) from the PI website. The link is in the "A000T0081-Downloading Manuals from PI.pdf" file in the \Manuals folder on the data storage device with the PI Software Suite.

### Updating the PC software and positioner database in Windows

- Use the PIUpdateFinder:
  - Follow the instructions in the manual for the PIUpdateFinder (A000T0028).

### Updating the PC software on 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. For **PI Software Suite C-990.CD1**: Select **ADD TO LIST+**
4. Select **REQUEST**
5. Fill out the download request form and send the request.  
The download link will be sent to the email address entered in the form.
6. Unpack the archive file on your PC to a separate installation directory.
7. In the directory with the unpacked files, go to the **linux** subdirectory.
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. 191).

### Updating the positioner database in Linux

1. Contact the customer service department (p. 191) to get the latest version of the **pistages2** or **pimicosstages2** 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.

### 5.6.3 Installing a Custom Positioner Database

With a custom positioner, you will receive, if necessary, a file from PI with a custom positioner database. You have to install this file on your PC so that you can load the parameter values for the custom positioner in the C-877.

#### Installing a custom positioner database on Windows

1. Open the \PI\GCSTranslator directory on your PC:

If you work with PIMikroMove:

- a) Go to the PIMikroMove main window and open the **Version Information** window via the **Connections > Search for controller software** menu item.
- b) In the **Version Information** window, click on the **Show GCS PATH...** button to open the \PI\GCSTranslator directory in Windows Explorer.

The path in which the \PI directory is located was determined during the installation of the PC software, normally C:\ProgramData.

2. Copy the positioner database file to the \PI\GCSTranslator directory on your PC.

#### INFORMATION

If the \PI\GCSTranslator directory is not present on your PC:

For an executable file (.exe) to be able to access a positioner database, both files have to be in the same directory.

#### Installing a custom positioner database on Linux

1. Log into the PC as superuser (root privileges).
2. Copy the positioner database file to the /usr/local/PI/pi\_gcs\_translator/ directory.

## 5.7 Connecting the PC

Communication between the C-877 and a PC is required to configure the C-877 and to command motions using the GCS commands. The C-877 is also equipped with USB interface.

In this section, you learn how to establish the corresponding cable connections between the C-877 and a PC.

The steps for establishing communication between C-877 and the PC are described in section "Startup": "Establishing Communication via USB" (p. 50)

### 5.7.1 Connecting the C-877 via the USB interface

#### Requirements

- ✓ The PC has a free USB interface.

#### Tools and accessories

- USB cable (type A to Mini-B) for connecting to the PC (000014651; in the scope of delivery)

#### Connecting the C-877 to the PC

- Connect the USB socket of the C-877 and the USB interface of the PC with the USB cable.



## 6 Startup

### 6.1 General Notes on Startup

#### NOTICE



#### Damage due to disabled limit switch evaluation!

The collision of a moving part at the end of the travel range, or with an obstacle, as well as high acceleration, can cause damage to or considerable wear on the mechanics.

- Avoid motion in open-loop operation.
- If motion in open-loop operation is necessary:
  - Set the control value with the SMO command so that the axis moves with low velocity.
  - Stop the axis in time. For this purpose, use the #24, STP or HLT command, or set the control value to zero with the SMO command.
- Do not disable the evaluation of the limit switches by the C-877 via parameter setting.
- Check the function of the limit switches at about 10 % to 20 % of the maximum velocity.
- In the event of a malfunction of the limit switches, stop the motion immediately.

### 6.2 Switching the C-877 On

#### INFORMATION

The C-877 is intended for closed-loop operation with incremental position sensors (servo mode On). After switch-on, open-loop operation is enabled by default (servo mode Off).

- Query the current operating mode with the `SVO?`, `#4` or `SRG?` commands.
- Enable closed-loop operation with the `SVO` command.
- If necessary, program a startup macro that starts the C-877 via the `SVO` command in closed-loop operation; see "Setting up a startup macro" (p. 71).

#### INFORMATION

The ID chip is not read when you connect the positioner while the C-877 is switched on.

- After connecting a positioner, reboot the C-877 with the `RBT` (p. 117) command or with the corresponding PC software functions in order to read the data from the ID chip.

#### Requirements

- ✓ You have read and understood the General Notes on Startup (p. 49).
- ✓ The C-877 has been installed properly (p. 41).

**Switching on the C-877**

- Connect the power cord of the power supply with the power socket.  
The C-877 loads information to the volatile memory in the following order:
  - a) Parameter values from the nonvolatile memory
  - b) Parameter values from the ID chip of the positioner
 The **STA** LED on the front panel of the C-877 displays the state of the C-877:
  - green: C-877 is ready for normal operation
  - off: The C-877 is not connected to the power supply or could be defective
- If the C-877 is properly connected to the power supply (p. 43) and the **STA** LED does not light up after you switch on the power, contact our customer service department (p. 191).

**6.3 Establishing Communication****INFORMATION**

A USB UART module (FTDI) is used for the USB interface in the C-877. Therefore, if the C-877 is connected via USB and switched on, the USB interface is also shown as COM port in the PC software. The C-877 uses a baud rate of 115200 for this interface.

The procedure for PIMikroMove is described in the following.

For information on establishing the communication on Linux systems see the Technical Note "PI Software on ARM-Based Platforms", A000T0089 (p. 3).

**6.3.1 Establishing Communication via the USB Interface****INFORMATION**

If the controller is connected via the USB connection and switched on, the USB interface in the PC software is also shown as a COM port.

**Requirements**

- ✓ You have read and understood the General Notes on Startup (p. 49).
- ✓ The C-877 is connected to the USB interface of the PC (p. 47).
- ✓ The C-877 is switched on (p. 49).
- ✓ The PC is switched on.
- ✓ The required software and USB drivers are installed on the PC (p. 44).
- ✓ You have read and understood the manual of the PC software used. The links to the software manuals are in the A000T0081 file on the PI software data storage medium.

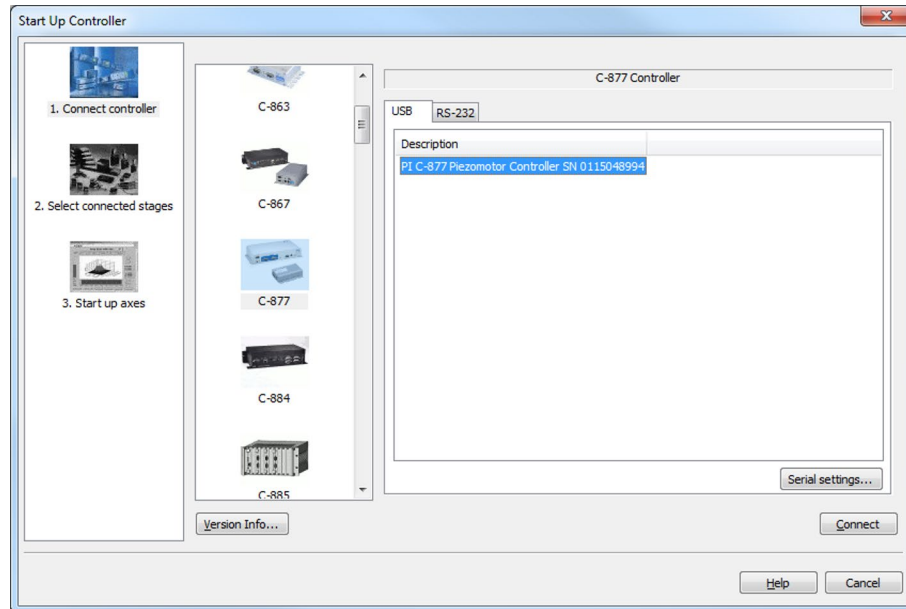


### Establishing communication via USB

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-877** in the field for controller selection.
3. Select the **USB** tab on the right-hand side of the window.
4. On the **USB** tab, select the **C-877** connected.
5. Click **Connect** to establish communication.

If communication was established successfully, PIMikroMove guides you through the configuration of the C-877 for the connected positioner; see "Starting Motion" (p. 51).

- If communication could not be established, look for a solution to the problem in "Troubleshooting" (p. 187).

## 6.4 Starting Motion

PIMikroMove is used in the following to move the positioner. The program guides you through the following steps so that you do not have to deal with the respective GCS commands:

- Configuration of the C-877 for the connected positioner
- Switching servo mode on
- Refer to "Referencing" (p. 34) for details on doing a reference move.

It is then possible to run the first motion tests for the positioner.

**NOTICE****Selecting an incorrect positioner type**

Selecting an incorrect positioner type in the PC software can damage the positioner.

- Make sure that the type of positioner selected in the PC software matches the positioner that is connected.

**NOTICE****Oscillation!**

Unsuitable setting of the C-877's servo control parameters can cause the positioner to oscillate. Oscillation can damage the positioner and/or the load fixed to it.

- Secure the positioner and all loads adequately.
- If the connected positioner is oscillating (unusual operating noise), immediately switch off the servo mode or disconnect the C-877 from the power source.
- Only switch on the servo mode after you have modified the servo control parameter settings of the C-877; see "Optimizing Servo Control Parameters" (p. 56).
- If, due to a very high load, oscillation occurs during the reference move, follow the instructions for the reference move in "Troubleshooting" (p. 187).

**INFORMATION**

After communication has been established between the C-877 and the PC, PIMikroMove guides you through the configuration of the C-877 for the connected positioner. Selection of the configuration steps offered by PIMikroMove is based on evaluation of the following parameter values in the volatile memory of the C-877:

- **Stage Name** (ID 0x3C): The value is used by PIMikroMove as a criterion for finding a suitable parameter set in the positioner databases.
- **Stage Type** (ID 0x0F000100): The value was loaded from the ID-Chip (p. 11) of the connected positioner when the C-877 is switched on.

Possible configuration steps:

- When the values of the parameters 0x3C and 0x0F000100 are identical, PIMikroMove assumes that all parameters of the C-877 have already been adapted to the connected positioner. The **Start up controller** window goes directly to the **Start up axes** step, where the reference move can be started.
- If the values of the parameters 0x3C and 0x0F000100 are not identical, the **Stage Type Configuration** window opens. The **Yes, configure for ...** button can be used to load a suitable parameter set from a positioner database to the C-877. After the parameter set has been loaded, the **Start up controller** window goes to the **Start up axes step**. If a matching parameter set is not in the positioner databases, a corresponding notice will appear in the **Stage Type Configuration** window.
- If the value of the parameter 0x0F000100 is empty because the positioner does not have an ID chip, for example, the **Start up controller** window will go to the **Select connected stages** step.

**Requirements**

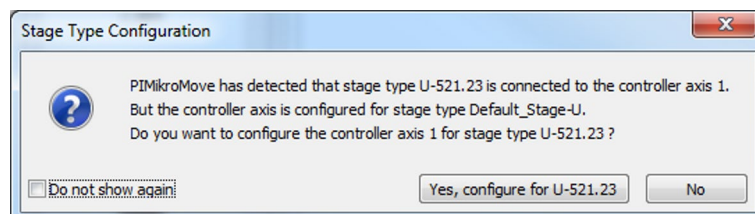
- ✓ You have read and understood the General Notes on Startup (p. 49).

- ✓ PIMikroMove is installed on the PC (p. 44).
- ✓ You have read and understood the PIMikroMove manual. The links to the software manuals are in the file A000T0081 on the data storage device with the PI software.
- ✓ You have installed the latest version of the positioner database(s) onto your PC (p. 45).
- ✓ If PI has provided you with a custom positioner database for your positioner, then you have installed this database on your PC (p. 46).
- ✓ You have installed the positioner in the same way as it will be used in your application (corresponding load, orientation, and mounting).
- ✓ You have connected the C-877 to the positioner (p. 46).
- ✓ You have established communication with PIMikroMove between the C-877 and the PC (p. 50).

### Starting motion with PIMikroMove

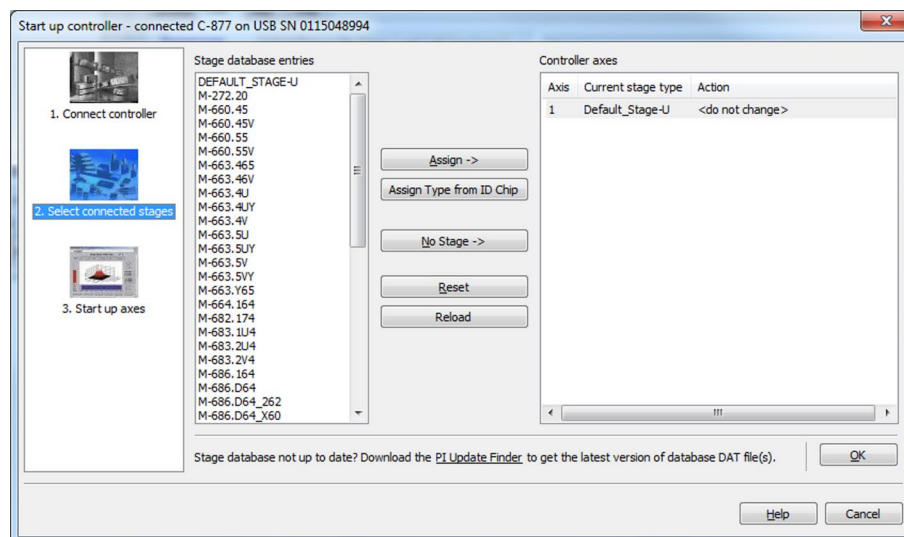
1. If one of the two following points applies, configure the C-877 for the connected positioner:
  - The **Stage Type Configuration** dialog has opened.
  - The **Select connected stages** step is displayed in the **Start up controller** window.

If the **Stage Type Configuration** dialog has opened:



- Click the **Yes, configure for ...** button to load the appropriate parameter set from the positioner database into the C-877. This opens the **Save all changes permanently?** dialog.

If the **Select connected stages** step is displayed in the **Start up controller** window:

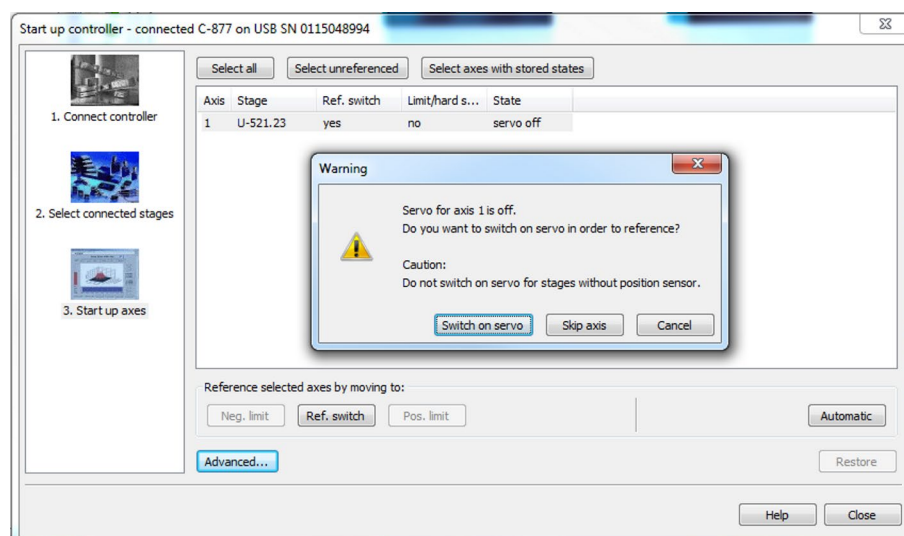


- a) Select the matching positioner type. You have two options:
  - Click **Assign Type from ID Chip**.
  - Mark the appropriate positioner type in the **Stage database entries** list and click **Assign**.
- b) Confirm selection with **OK** to load the parameter settings for the selected positioner type from the positioner database into the C-877. This opens the **Save all changes permanently?** dialog.
2. Specify how you want to load the parameter settings into the C-877 in the **Save all changes permanently?** dialog box:
  - Temporary load: Click **Keep the changes temporarily** to load the parameter settings into the volatile memory of the C-877. The settings are lost when the C-877 is switched off or rebooted.
  - Load as default values: Click **Save all settings permanently on controller** to load the parameter settings into the nonvolatile memory of the C-877. The settings are available immediately after switching on or rebooting the C-877 and do not need to be reloaded.

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

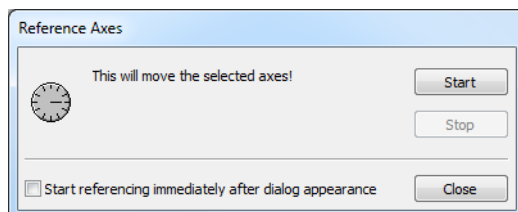
3. During the **Start up axes** step, do a reference move for the axis so that the controller knows the absolute axis position: You have the following options (options not supported by the positioner/controller either do not exist or cannot be activated):
  - If you want to start the reference move to the reference switch, click **Ref. switch**.
  - If you want to start the reference move to the negative limit switch, click **Neg. limit**.
  - If you want to start the reference move to the positive limit switch, click **Pos. limit**.

If a warning message appears indicating that servo mode is switched off:



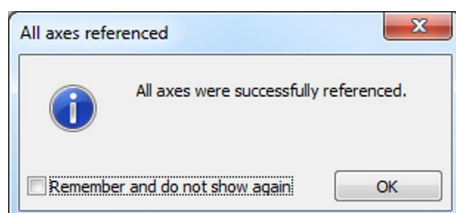
- a) Switch on the servo mode by clicking on the **Switch on servo** button.

If the **Reference Axes** dialog is displayed after switching on the servo:



b) Click the **Start** button. The axis performs the reference move.

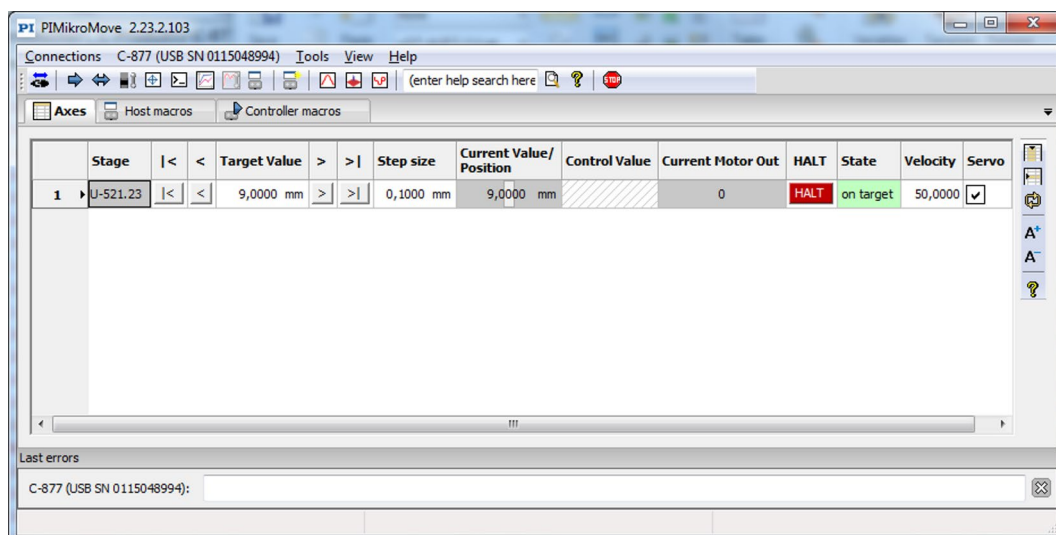
If the corresponding message is displayed after a successful reference move:



c) Close the message with **OK**.

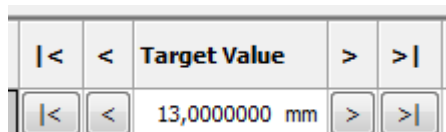
- After a successful reference move, close the **Start up controller** window by clicking **Close**.

The main window of PIMikroMove opens.



- Test the motion of the axis several times.

By clicking the corresponding arrow keys for the axis in the main window of PIMikroMove for example, it is possible to initiate motion over a particular distance (specification in **Step size** column) or to the limits of the travel range.



## 6.5 Optimizing the Servo Control Parameters

The dynamic characteristics of the system (e.g., starting motion, oscillation behavior during motion, overshoot, and settling time) are optimized by adjusting the parameters for servo algorithm and corrections to the dynamics profile. The optimum settings depend on your application and your wishes.

Typically, optimization is determined empirically, i.e., various values are used in closed-loop operation and the behavior of the positioner is monitored. Optimization is done with the following parameters:

- P, I, D terms and I limit of parameter groups 0 to 4 (IDs 0x4n1, 0x4n2, 0x4n3, 0x4n4; n takes on a value of 0 to 4 depending on the parameter group)
- Parameters for switching between parameter groups 0 to 4, e.g., window limits (IDs 0x4n6, 0x4n7; n takes on a value from 0 to 4 depending on the parameter group)
- Velocity-dependent offset and offsets for the positive and negative direction of motion (IDs 0x48, 0x33, 0x34)

### INFORMATION

Experienced users can also enable the two-phase control (p. 20) to improve the settling behavior.

The following describes the procedure for optimizing the dynamic characteristics of the system for PIMikroMove. Two-phase control is **not** taken into account here.

### Requirements

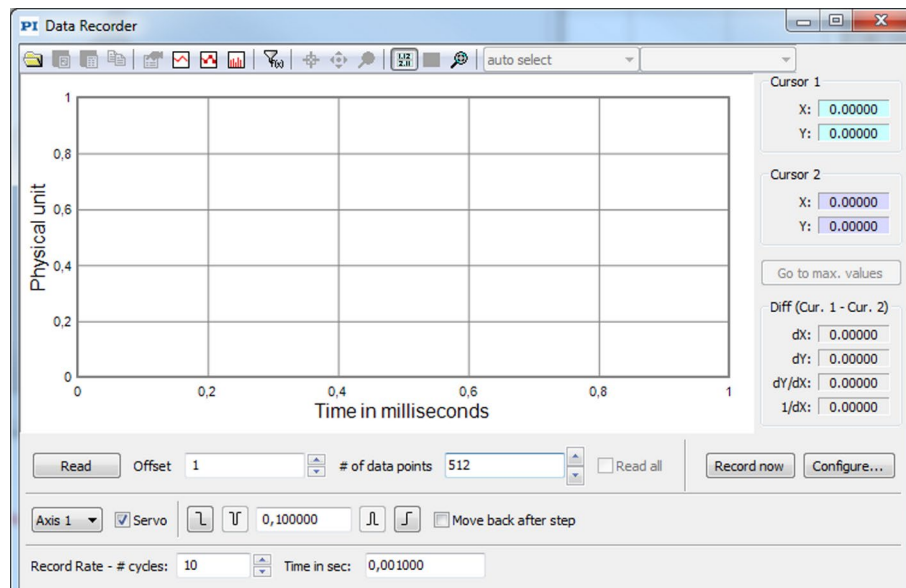
- ✓ You have installed the positioner in the same way as it will be used in your application (corresponding load, orientation and fixing).
- ✓ You started initial motion (p. 51) with PIMikroMove.
- ✓ All devices are still ready for operation.

### Checking the servo control parameters: Measuring the step response

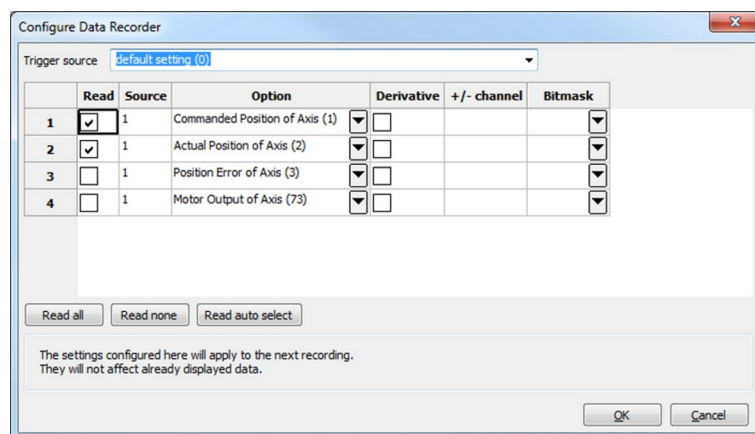
You determine the settling behavior of the positioner in closed-loop operation with the recording of the step response.

1. Open the **Data Recorder** window in the main window of PIMikroMove via the **C-877 > Show data recorder** menu item.
2. Switch servo mode on by clicking the **Servo** checkbox (check).
3. Configure the data recorder.
  - a) For the amplitude of the step to be carried out, set a value that is typical for your application, e.g., 0.100000 (specified as physical unit, e.g., millimeters or degrees).
  - b) Set the value 10 for the record table rate in the **Record Rate - # cycles** field.

- c) Set the value 1024 (or less) for the number of data points to be read for the graphic display in the **# of data points** field.



- d) Click the **Configure...** button and make sure that "Commanded Position of Axis" and "Actual Position of Axis" are selected in the **Configure Data Recorder** window as the variables to be recorded. Close the window with **OK**.





	Read	Source	Option	Derivative	+/- channel	Bitmask
1	<input checked="" type="checkbox"/>	1	Commanded Position of Axis (1)	<input type="checkbox"/>		
2	<input checked="" type="checkbox"/>	1	Actual Position of Axis (2)	<input type="checkbox"/>		
3	<input type="checkbox"/>	1	Position Error of Axis (3)	<input type="checkbox"/>		
4	<input type="checkbox"/>	1	Motor Output of Axis (73)	<input type="checkbox"/>		

Read all Read none Read auto select

The settings configured here will apply to the next recording.  
They will not affect already displayed data.

OK Cancel

4. Click the  button in the **Data Recorder** window to start the step in the positive direction as well as the recording.
- The axis performs the step and the step response is recorded and displayed graphically.
5. Check the displayed step response (see example below).
- If necessary, enlarge the view by clicking the  button and, while pressing the left mouse button, dragging the mouse pointer, which has turned into a magnifying glass, over a section of the graphic display (clicking the right mouse button in the graphics field reduces the view back to the original size).

Examples for step responses:

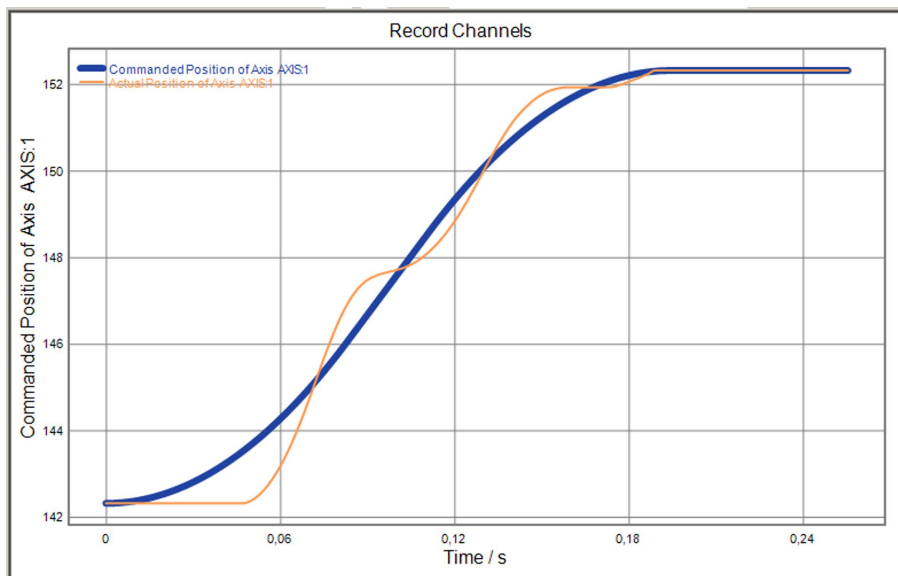


Figure 12: Improper settings, causing oscillation and unacceptable settling behavior

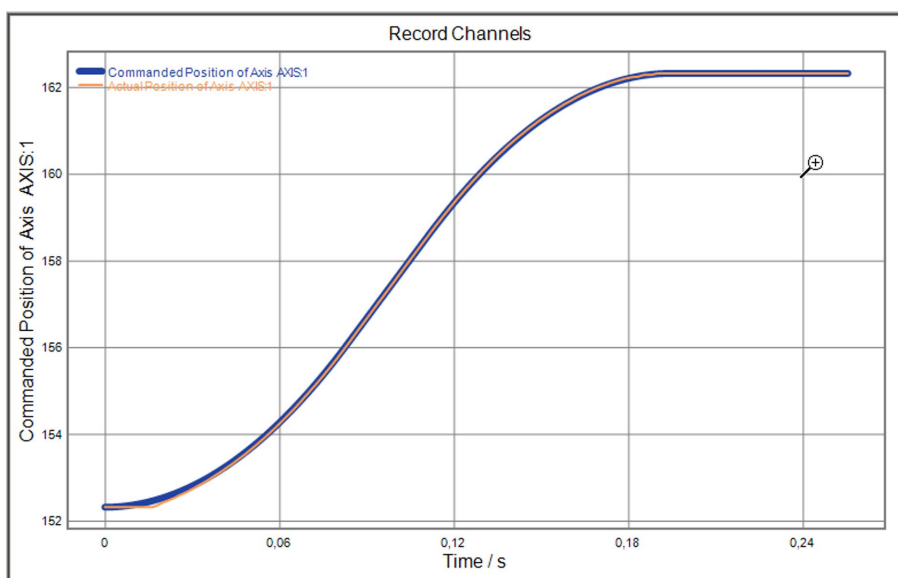


Figure 13: Settling behavior already almost optimum, but behavior at start of the motion not yet satisfactory (offset settings still have to be optimized)



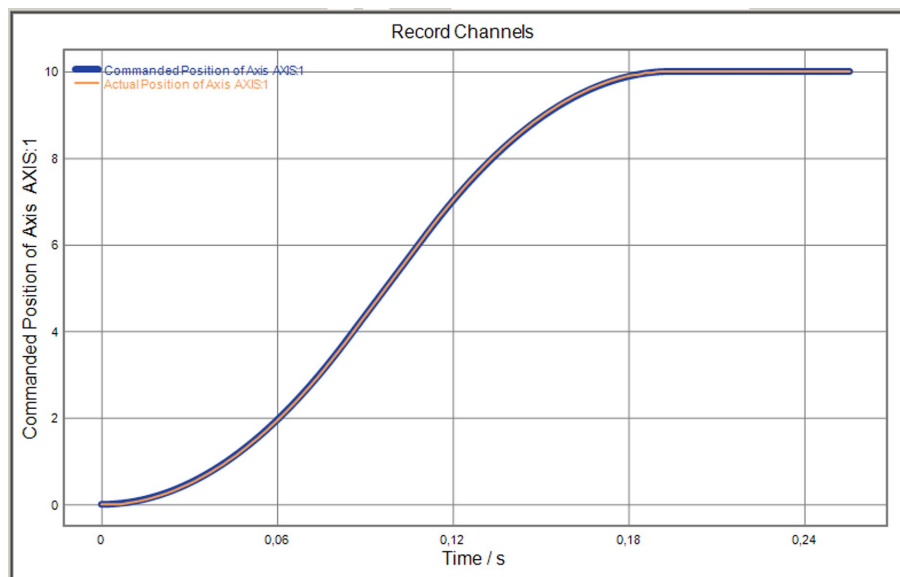


Figure 14: Optimum dynamic characteristics of the system, no adjustments necessary

If the result is satisfactory:

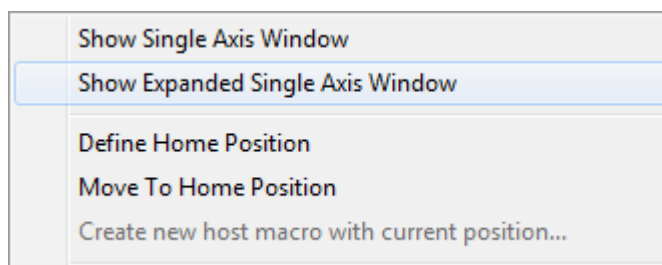
- You already have optimum parameter settings and do not have to do anything further.

If the result is not satisfactory:

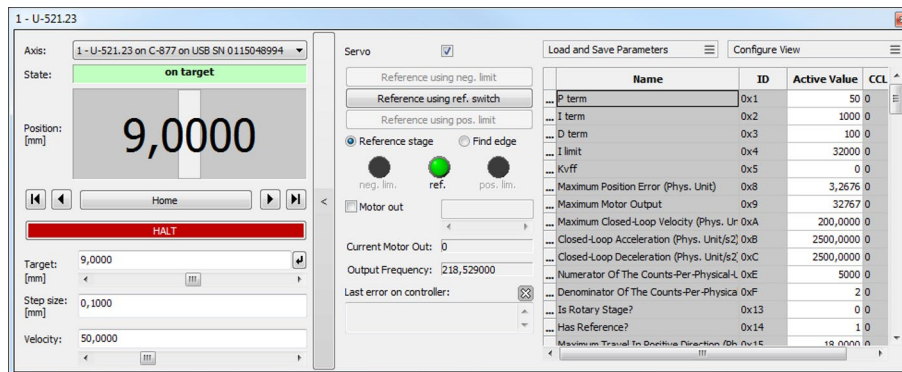
- Optimize the parameters for the dynamic characteristics of the system; see below.

### Optimizing the servo control parameters

1. Open the expanded single axis window for the connected positioner in the main window of PIMikroMove by clicking the right mouse button on the corresponding line of the **Axes** tab and selecting **Show Expanded Single Axis Window** in the context menu.



2. Enter new values for the parameters to be adapted:



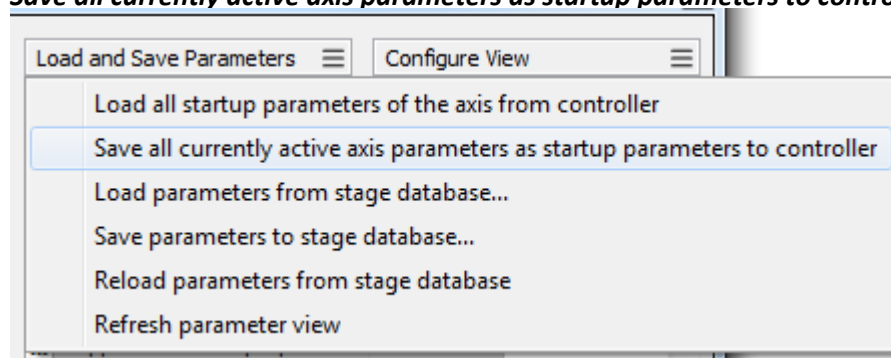
- If the parameter to be modified is not included in the list on the right-hand side of the window, click on **Configure view > Select parameters...** and add it to the list. You can also display certain groups of parameters or all axes-related parameters.
  - Type the new parameter value into the corresponding input field in the **Active Value** column of the list.
  - Press the **Enter** key on the PC keyboard or click outside the input field with the mouse to transfer the parameter value to the volatile memory of the controller. Note: If a parameter value in the volatile memory (**Active Value** column) is different to the parameter value in the nonvolatile memory (**Startup Value** column), the line in the list is highlighted in color.
3. In the **Data Recorder** window, record the step response of the positioner again.

If the result is not satisfactory:

- Enter different values for the servo control parameters and record the step response again.

If you are satisfied with the result and want to keep the new servo control parameter settings, save the new settings. You have the following options:

- Save a parameter set in the positioner database on the PC by clicking on **Load and Save Parameters -> Save parameters to stage database...**, see "Creating or Modifying a Positioner Type" (p. 168).
- Transfer the current values of the listed parameters from the volatile memory to the non-volatile memory of the C-877 by clicking **Load and Save Parameters -> Save all currently active axis parameters as startup parameters to controller**.



## 7 Operation

### 7.1 Protective Functions of the C-877

#### 7.1.1 Protection Against Overheating

When a high control value remains set over a long period of time, the connected positioner can heat up. Overheating can result in damage to the positioner.

The **PID Maximum Output Time (s)** parameter (ID 0x7B) specifies the maximum time period for which a high control value may be set in closed-loop operation. A high control value is present when the following applies:

Current absolute measure of the control value of the control value  $\geq 95\%$  of **Maximum Motor Output** (ID 0x9).

If the high control value is still set after the maximum time period has expired, the C-877 will react as follows to protect the system against damage:

- The control value is set to the value zero for the axis in question.
- The servo mode is switched off for the axis in question.

#### 7.1.2 Behavior with Motion Errors

There is motion error if the current position of the axis does not change when the control value changes.

Motion error is possible in closed-loop or open-loop operation.

In closed-loop operation, a motion error occurs when the difference between the actual and the commanded position exceeds the specified maximum. The range in which the deviation may be, is defined by the parameter **Maximum Position Error (Phys. Unit)** (0x8).

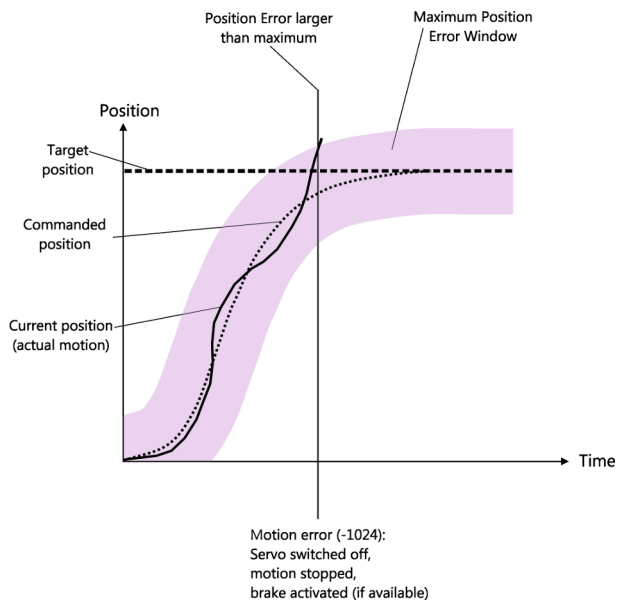
Motion errors can have the following causes, for example:

- Drive malfunction
- Position sensor malfunction
- Positioner malfunction

If there is a motion error, the C-877 reacts as follows to protect the system against damage:

- Servo mode is switched off for the axis in question.
- Motion is stopped.
- Error code -1024 is output.

Then restore operational readiness for the C-877.



### 7.1.3 Behavior During System Errors

There is a system error when C-877 is not responsive.

For example, the cause of a system error can be a buffer overflow in the firmware of the C-877.

If a system error occurs the C-877 reacts as follows:

- After a certain delay, the safety function of the **Watchdog Timer** initiates a reboot of the C-877.

### 7.1.4 Re-establishing Readiness for Operation

1. Send the `ERR?` command to read out the error code.  
The `ERR?` command resets the error code to zero during the query.
2. Check your system and make sure that the following points are fulfilled:
  - The axis can be moved without danger.
  - The C-877 has **not** overheated (internal temperature is maximum 65 °C).
3. If the servo mode was switched off after an error or overheating:
  - Switch on the servo mode for the axis with the `SVO` command.

When the servo mode is switched on, the target position is set to the current axis position.

## 7.2 Data Recorder

### 7.2.1 Configuring the Data Recorder

The C-877 contains a real-time data recorder. The data recorder can record for example, the current position of the axis.

The recorded data is stored temporarily in 4 data recorder tables with 1024 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.

#### **INFORMATION**

The settings for configuring the data recorder can only be changed in the volatile memory of the C-877. After switching on or rebooting the C-877, the factory settings are active, unless a configuration has already been made using a startup macro.

#### **Reading general information from the data recorder**

- Send the `HDR?` command (p. 104).

The available options are displayed for recording and triggering, as well as the information on additional parameters and commands for data recording.

#### **Configuring data to be recorded**

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

- Send the `DRC?` command (p. 95) 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-877 record the following:
  - Data recorder table 1: Record option 1: Commanded position of the axis
  - Data recorder table 2: Record option 2: Current position of the axis
  - Data recorder table 3: Record option 3: Position error of the axis
  - Data recorder table 4: Record option 73: Control value of the axis
- Configure the data recorder with the `DRC` command (p. 93).

#### **Configuring the recording trigger**

You can specify how the recording is to be triggered.

- Get the current trigger option with `DRT?` (p. 98).
- Change the trigger option with the `DRT` command (p. 97). The trigger option applies to all data recorder tables whose record option is not set to 0.

#### **Setting the record table rate**

- Send the `RTR?` command (p. 121) to read out the record table rate of the data recorder.

The parameter indicates the number of servo cycles that are required for recording each data point. The default value is 10 servo cycles. The servo cycle time of the C-877 is 100 µs.

- Change the record table rate with the `RTR` command. (p. 120)

As the record table rate increases, you increase the maximum duration of the data recording.

## 7.2.2 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 when a step response measurement is started with `STE` (p. 130).

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.

## 7.2.3 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. 95).  
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. 95).

## 7.3 Controller Macros

### 7.3.1 Overview: Macro Functionality and Example Macros

The C-877 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-877 is switched on or rebooted.
- Processing a macro and stopping macro execution can be linked to conditions. In this way, loops can be realized as well.
- Macros can call up themselves or other macros in several nesting levels.
- Variables (p. 78) can be set for the macro and in the macro itself and used in different operations.

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

- Moving an axis back and forth (p. 68)
- Moving an axis with a variable travel range back and forth (p. 70)
- Preparing an axis via startup macro for closed-loop operation (p. 71)

### 7.3.2 Commands and Parameters for Macros

#### Commands

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

Command	Syntax	Function
ADD (p. 87)	ADD <Variable> <FLOAT1> <FLOAT2>	Adds two values and saves the result to a variable (p. 78). Can only be used for local variables in macros.
CPY (p. 89)	CPY <Variable> <CMD?>	Copies a command response to a variable (p. 78). Can only be used for local variables in macros.
DEL (p. 91)	DEL <uint>	Can only be used in macros. Delays <uint> milliseconds.
JRC (p. 107)	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. 108)	MAC BEG <macro name>	Starts the recording of a macro with the name <i>macro name</i> on the controller. <i>macro name</i> can consist of up to 8 characters.
	MAC DEF <macro name>	Defines the specified macro as the startup macro.
	MAC DEF?	Gets the startup macro.
	MAC DEL <macro name>	Deletes the specified macro.
	MAC END	Ends the macro recording.
	MAC ERR?	Reports the last error that occurred while the macro was running.
	MAC NSTART <macro name> <uint> [<String1> [<String2>]]	Starts the specified macro n times in succession (n = number of times). The values of local variables can be set for the macro with <String1> and <String2>.
	MAC START <macro name> [<String1> [<String2>]]	Runs the specified macro. The values of local variables can be set for the macro with <String1> and <String2>.
MAC ? (p. 111)	MAC? [<macro name>]	Lists all macros or the content of a specified macro.
MEX (p. 113)	MEX <CMD?> <OP> <Value>	Can only be used in macros. Stops the macro execution depending on a condition.

Command	Syntax	Function
RMC? (p. 118)	RMC?	Lists macros which are currently running.
VAR (p. 135)	VAR <Variable> <String>	Sets a variable (p. 78) to a certain value or deletes it. Can only be used for local variables in macros.
VAR? (p. 135)	VAR? [{<Variable>}]	Gets variable values.
WAC (p. 138)	WAC <CMD?> <OP> <Value>	Can only be used in macros. Waits until a condition is met.
#8 (p. 84)	-	Tests if a macro is running on the controller.

### Parameters

The following parameter is available for working with macros:

Parameters	Description and Possible Values
<b>Ignore Macro Error?</b> 0x72	Determines whether the controller macro is stopped if an error occurs when it is running. <ul style="list-style-type: none"> <li>0 = Stop macro when error occurs (default)</li> <li>1 = Ignore error</li> </ul>

### 7.3.3 Working with Macros

Work with macros comprises the following:

- Recording macros (p. 67)
- Starting macros (p. 69)
- Stopping macros (p. 71)
- Configuring a startup macro (p. 71)
- Deleting macros (p. 72)

#### INFORMATION

The C-877 can save up to 32 macros simultaneously. A maximum of 5 nesting levels are possible in macros.

#### INFORMATION

Basically all GCS commands (p. 75) can be included in a macro. Exceptions:

- RBT for rebooting the C-877
- MAC BEG and MAC END for macro recording
- MAC DEL for deleting a macro



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 send 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. 78).

### 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. 78) in macros to make macros more flexible, and give the corresponding variable values when starting macro execution.
- Contact our customer service department (p. 191) if the C-877 shows unexpected behavior.

### INFORMATION

A macro is overwritten if a macro with the same name is re-recorded.

### 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

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 on 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 up themselves or other macros in 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 on 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 has been stored in the nonvolatile memory of the C-877.

4. If you want to check whether the macro has been correctly recorded:

If you are working with PITerminal or in the **Command entry** window of PIMikroMove:

- Get which macros are saved in the C-877 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 on 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 commands `MAC BEG` and `MAC END` must be left out.

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

## Starting a macro

### INFORMATION

Any commands can be sent from the command line when a macro is running on the controller. The macro content and motion commands received from the command line can overwrite each other.

### 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, PITerminal or the **Command entry** window of PIMikroMove is used to enter commands. Details on working with the **Controller macros** tab in PIMikroMove are 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. 163).

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 commands `MAC`, `BEG` and `MAC END` must be left out.

The axis 1 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-877 is switched on or rebooted, since they are only written to the volatile memory of the C-877.

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 1 $1
```

```
WAC ONT? 1 = 1
```

```
MAC END
```

MOVWAI moves axis 1 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 1 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, PITerminal or the **Command entry** window of PIMikroMove is used to enter commands. Details on working with the **Controller macros** tab in PIMikroMove are 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 macro execution, send the `MAC ERR?` command. The response shows the last error that occurred.

## Configuring a startup macro

Any macro can be defined as the startup macro. The startup macro is executed each time the C-877 is switched on or rebooted.

### INFORMATION

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

In the following, PITerminal or the **Command entry** window of PIMikroMove is used to enter commands. Details on working with the **Controller macros** tab in PIMikroMove are 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: Preparing an axis via a startup macro for closed-loop operation

### INFORMATION

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

The `STARTCL` macro switches servo mode for axis 1 on and starts a reference move to the negative limit switch. As `STARTCL` is defined as the startup macro, axis 1 is ready for closed-loop operation immediately after switch-on.

- Send:
 

```
MAC BEG startcl
SVO 1 1
```

```
DEL 1000
FNL 1
MAC END
MAC DEF startcl
```

### INFORMATION

When using this macro, the parameter settings of the C-877 should be adapted in the nonvolatile memory to the positioner connected. Alternatively, the parameter settings can also be configured in the volatile memory via the startup macro. For further information, see "Adapting Settings" (p. 163).

### Deleting a macro

### INFORMATION

A macro cannot be deleted while it is running.

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

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

## 7.3.4 Making Backups and Loading Controller Macros

For example, making backups of controller macros on the PC can be useful before updating the firmware (p. 183).

### INFORMATION


The use of the **Controller macros** tab in PIMikroMove is recommended for backing up and loading controller macros. A detailed description of the tab can be found in the PIMikroMove manual.

### Backing up controller macros onto the PC with PIMikroMove

1. Select the **Controller macros** tab in the PIMikroMove main window.
2. Select the macros in the **Macros on controller** list that you want to back up to the PC:
  - Click the desired entry in the list to select an macro.
  - To select several macros, hold down the Shift button and click the desired entries in the list.
  - To deselect, click an empty area in the list.


By selecting one or more macros, the  (*Save selected macros to PC*) button becomes active.

3. Save the selected macros on the PC:

- a) Click the  button to open a directory selection window.
- b) Select the directory on the PC where you want to save the macros.
- c) Click **Save**.

The macros are saved as text files (<macro name>.txt) to the directory selected on the PC.

### Loading controller macros from the PC to the C-877 with PIMikroMove

1. Select the **Controller macros** tab in the PIMikroMove main window.
2. Load macros from the PC to the C-877:
  - a) Click the  button to open a file selection window.
  - b) Select the text files (<macro name>.txt) in the file selection window whose contents you want to load as a macro from the PC to the C-877.
  - c) Click **Open**.

For each selected text file (<macro name>.txt), the content is loaded as a macro <macro name> into the C-877.





## 8 GCS Commands

### 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.
{...}	Curly brackets indicate a repetition of entries, i.e., it is possible to access more than one element (e.g., several axes) in one command line.
<b>LF</b>	LineFeed (line feed, ASCII character 10), is the default termination character (character at the end of a command line).
<b>SP</b>	Space (ASCII character 32) indicates a space.
"..."	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 has to be terminated with a line feed (**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. 13) for the identifiers supported by the C-877.

Example 1:

Axis 1 is to be moved to position 10.0. The unit depends on the controller (e.g.,  $\mu\text{m}$  or mm).

Send:     MOV SP1 SP10.0 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 SP1 SP17.3 SP2 SP2.05 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?****SP**2**SP**1**LF**

Receive: 2=-1158.4405**SPLF**

1=+0000.0000**LF**

### INFORMATION

With the C-877 only a single element per command line can be addressed (e. g. axis or parameter).

Example:

By sending command line

**SEP** 100 1 0x32 0

a new value of parameter 0x32 is saved in nonvolatile memory for axis 1,  
sending command line

**SEP** 100 1 0x32 0 1 0x14 1

is not possible, however, because two parameters are to be changed.

If the command supports this, all elements can be addressed by omitting the element identifier.

Example:

By sending command line

**SEP?**

all parameters from the nonvolatile memory are queried.

## 8.3 Target and Sender Address

In principle, the addresses of the target controller and the sender are required in every command line. This applies even to single-character commands like #4 or to macro recording.

Because the PC may only send command lines to the controller, the sender address (0) can be omitted. However, both the target and the sender addresses are part of the controller reply. Multiline responses include the target and sender address only in the first line.

Because the target address can be omitted when the target controller has address 1, the target address (1) can be omitted. If the target address is omitted when addressing a controller, the target and sender addresses will also be omitted in the reply of the controller.

Example: Query the device identification string of the C-877.1U11 with address 1

Send: **\*IDN?**

The controller replies:

```
(c) 2024 Physik Instrumente (PI) Karlsruhe, C-877.1U11, 0, 1.2.0.0
```

Send: `1 *IDN?`

The same controller replies:

```
0 1 (c) 2024 Physik Instrumente (PI) Karlsruhe, C-877.1U11, 0, 1.2.0.0
```

## 8.4 Variables

For more flexible programming, the C-877 supports variables. While global variables are always available, local variables are only valid for a specified macro. Typically, variables are used when working with macros.

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

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.

### Local variables:

- Local variables can only be used in macros.
- At present, the controller firmware supports three local variables: 0, 1 and 2.
- The values of the local variables 1 and 2 are given as arguments of the `MAC START` or `MAC NSTART` command when starting the macro.

The command formats are:

```
MAC START <macroname> [<String1> [<String2>]]
```

```
MAC NSTART <macroname> <uint> [<String1> [<String2>]]
```

<STRING1> and <STRING2> give the values for the local variables 1 and 2 used in the macro. <STRING1> and <STRING2> can be given directly or via the values of variables.

<uint> gives the number of times the macro is to be run. Further information can be found in the description of the `MAC` command (p. 108).

- 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.
- Within a macro, the values of local variables can be modified using the `ADD` (p. 87), `CPY` (p. 89), `MAT` (p. 112), and `VAR` (p. 135) commands and can be deleted with the `VAR` command (exception: Local variable 0).

- As long as the macro is running, the values of the local variables can be queried with

```
VAR? 0
```

```
VAR? 1
```

```
VAR? 2
```

The queries can be sent inside or outside of the macro.

#### Global variables:

- Global variables can be used inside and outside of macros.
- The maximum number of global variables is 10.
- Global variables are created and modified using the `ADD`, `CPY`, `MAT`, or `VAR` commands. They can be deleted with `VAR`.
- The variable values can be queried with `VAR?`.

## 8.5 Command Overview

Com-mand	Arguments	Description
#4		Request Status Register (p. 83)
#5		Request Motion Status (p. 83)
#7		Request Controller Ready Status (p. 84)
#8		Query If Macro Is Running (p. 84)
#24		Stop All Axes (p. 85)
*IDN?		Get Device Identification (p. 85)
ACC	{<AxisID> <Acceleration>}	Set Closed-Loop Acceleration (p. 85)
ACC?	[{<AxisID>}]	Get Closed-Loop Acceleration (p. 86)
ADD	<Variable> <FLOAT1> <FLOAT2>	Add and Save To Variable (p. 87)
CCL	<Level> [<PSWD>]	Set Command Level (p. 88)
CCL?		Get Command Level (p. 88)
CPY	<Variable> <CMD?>	Copy Into Variable (p. 89)

Com- mand	Arguments	Description
CST?	[[<AxisID>]]	Get Assignment Of Stages To Axes (p. 89)
CSV?		Get Current Syntax Version (p. 90)
DEC	{<AxisID> <Deceleration>}	Set Closed-Loop Deceleration (p. 90)
DEC?	[[<AxisID>]]	Get Closed-Loop Deceleration (p. 91)
DEL	<uint>	Delay The Command Interpreter (p. 91)
DFH	[[<AxisID>]]	Define Home Position (p. 91)
DFH?	[[<AxisID>]]	Get Home Position Definition (p. 93)
DRC	{<RecTableID> <Source> <RecOption>}	Set Data Recorder Configuration (p. 93)
DRC?	[[<RecTableID>]]	Get Data Recorder Configuration (p. 95)
DRL?	[[<RecTableID>]]	Get Number Of Recorded Points (p. 95)
DRR?	[<StartPoint> <NumberOfPoints> [[<RecTableID>]]]	Get Recorded Data Values (p. 95)
DRT	{<RecTableID> <TriggerSource> <Value>}	Set Data Recorder Trigger Source (p. 97)
DRT?	[[<RecTableID>]]	Get Data Recorder Trigger Source (p. 98)
ERR?		Get Error Number (p. 98)
FED	{<AxisID> <EdgeID> <Param>}	Find Edge (p. 99)
FNL	[[<AxisID>]]	Fast Reference Move To Negative Limit (p. 100)
FPL	[[<AxisID>]]	Fast Reference Move To Positive Limit (p. 102)
FRF	[[<AxisID>]]	Fast Reference Move To Reference Switch (p. 102)
FRF?	[[<AxisID>]]	Get Referencing Result (p. 103)
GOH	[[<AxisID>]]	Go To Home Position (p. 104)
HDR?		Get All Data Recorder Options (p. 104)
HLP?		Get List of Available Commands (p. 105)
HLT	[[<AxisID>]]	Halt Motion Smoothly (p. 105)
HPA?		Get List Of Available Parameters (p. 106)
JRC	<Jump> <CMD?> <OP> <Value>	Jump Relatively Depending On Condition (p. 107)
LIM?	[[<AxisID>]]	Indicate Limit Switches (p. 108)

Com- mand	Arguments	Description
MAC	<keyword> {<parameter>} In particular: BEG <macro name> DEF <macro name> DEF? DEL <macro name> END ERR? FREE? NSTART <macro name> <uint> [<String1> [<String2>]] START <macro name> [<String1> [<String2>]]	Call Macro Function (p. 108)
MAC?	[<macro name>]	List Macros (p. 111)
MAN?	<CMD>	Get Help String For Command (p. 112)
MEX	<CMD?> <OP> <Value>	Stop Macro Execution Due To Condition (p. 113)
MOV	{<AxisID> <Position>}	Set Target Position (p. 114)
MOV?	[[<AxisID>]]	Get Target Position (p. 114)
MVR	{<AxisID> <Distance>}	Set Target Relative To Current Position (p. 115)
ONT?	[[<AxisID>]]	Get On-Target State (p. 116)
POS	{<AxisID> <Position>}	Set Real Position (p. 116)
POS?	[[<AxisID>]]	Get Real Position (p. 117)
RBT		Reboot System (p. 117)
RMC?		List Running Macros (p. 118)
RON	{<AxisID> <ReferenceOn>}	Set Reference Mode (p. 118)
RON?	[[<AxisID>]]	Get Reference Mode (p. 119)
RPA	[[<ItemID> <PamID>]]	Reset Volatile Memory Parameters (p. 119)
RTR	<RecordTableRate>	Set Record Table Rate (p. 120)
RTR?		Get Record Table Rate (p. 121)
SAI	{<AxisID> <NewIdentifier>}	Set Current Axis Identifiers (p. 121)
SAI?	[ALL]	Get List Of Current Axis Identifiers (p. 121)
SEP	<Pswd> {<ItemID> <PamID> <PamValue>}	Set Nonvolatile Memory Parameters (p. 122)
SEP?	[[<ItemID> <PamID>]]	Get Nonvolatile Memory Parameters (p. 123)
SMO	{<AxisID> <ControlValue>}	Set Open-Loop Control Value (p. 124)
SMO?	[[<AxisID>]]	Get Control Value (p. 125)
SPA	{<ItemID> <PamID> <PamValue>}	Set Volatile Memory Parameters (p. 126)

Com-mand	Arguments	Description
SPA?	[[<ItemID> <PamID>]]	Get Volatile Memory Parameters (p. 128)
SRG?	{<AxisID> <RegisterID>}	Query Status Register Value (p. 129)
STE	<AxisID> <Amplitude>	Start Step And Response Measurement (p. 130)
STP		Stop All Axes (p. 130)
SVO	{<AxisID> <ServoState>}	Set Servo Mode (p. 131)
SVO?	[[<AxisID>]]	Get Servo Mode (p. 132)
TCV?	[[<AxisID>]]	Get Commanded Closed-Loop Velocity (p. 132)
TMN?	[[<AxisID>]]	Get Minimum Commandable Position (p. 132)
TMX?	[[<AxisID>]]	Get Maximum Commandable Position (p. 133)
TNR?		Get Number Of Record Tables (p. 133)
TRS?	[[<AxisID>]]	Indicate Reference Switch (p. 134)
TVI?		Tell Valid Character Set For Axis Identifiers (p. 134)
VAR	<Variable> <String>	Set Variable Value (p. 135)
VAR?	[[<Variable>]]	Get Variable Value (p. 135)
VEL	{<AxisID> <Velocity>}	Set Closed-Loop Velocity (p. 136)
VEL?	[[<AxisID>]]	Get Closed-Loop Velocity (p. 137)
VER?		Get Versions Of Firmware And Drivers (p. 137)
WAC	<CMD?> <OP> <Value>	Wait For Condition (p. 138)
WPA	<Pswd> [[<ItemID> <PamID>]]	Save Parameters To Non-Volatile Memory (p. 138)



## 8.6 Command Descriptions for GCS 2.0

### #4 (Request Status Register)

Description: Queries system status information.

Format: #4

Arguments: none

Response: The response is bit-mapped. See below for the individual codes.

Notes: This command is identical in function to SRG? (p. 129), but only one character is sent via the interface. Therefore #4 can also be used while the controller is performing time-consuming tasks.

For the C-877, the response is the sum of the following codes, in hexadecimal format:

Bit	15	14	13	12	11	10	9	8
Description	On-target state	Determines the reference value	In motion	Servo mode on	-	-	-	Error flag

Bit	7	6	5	4	3	2	1	0
Description	-	-	-	-	Axis is referenced	Positive limit switch	Reference point switch	Negative limit switch

Example:

Send: #4

Receive: 0x900A

= Binary format: **1001000000001010**

Note: The response is in hexadecimal format. It means (displayed in bold):

The axis is at the target position (On-Target-Status; bit 15),  
servo mode is switched on (bit 12),  
no errors occurred (bit 8),  
the axis has already been referenced (bit 3),  
the positioner is at the position of the reference switch (bit 1).

**#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.

**#7 (Request Controller Ready Status)**

Description:	Queries the controller for ready state (tests if controller is ready to do a new command).  Note: Use #5 (p. 83) instead of #7 to verify if motion has finished.
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., executing a referencing move)
Troubleshooting:	The response characters may be displayed 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

### #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. 130), but only one character is sent via the interface.

Format:            #24

Arguments:        None

Response:         None

Notes:            #24 stops all motion caused by motion commands (e.g., MOV (p. 114), MVR (p. 115), GOH (p. 104), STE (p. 130), SMO (p. 124)), commands for referencing (FNL (p. 100), FPL (p. 102), FRF (p. 102)) and macros (MAC (p. 108)). Also stops macro running.

After the axes are stopped, their target positions are set to their current positions.

HLT (p. 105) in contrast to #24 stops motion with specified deceleration with respect to system inertia.

### \*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

Notes:            With C-877, \*IDN? responds something like:

```
(c)2024 Physik Instrumente (PI) GmbH &
Co. KG, C-877, 113059603, 1.005
```

**ACC (Set Closed-Loop Acceleration)**

Description:	Sets acceleration of specified axes.  ACC can be changed while the axis is moving.
Format:	ACC {<AxisID> <Acceleration>}
Arguments:	<AxisID> is one axis of the controller  <Acceleration> is the acceleration value in physical units/s <sup>2</sup> .
Response:	None
Troubleshooting:	Illegal axis identifiers
Notes:	The ACC setting only takes effect when the specified axis is in closed-loop operation (servo mode ON).  The lowest possible value for <Acceleration> is 0.  ACC changes the value of the <b>Closed-Loop Acceleration (Phys. Unit/s<sup>2</sup>)</b> parameter (ID 0xB) in the volatile memory of the C-877. The parameter value can be stored as default with WPA (p. 138), for details see "Adapting Settings" (p. 163).  The maximum value that can be set with the ACC command is specified by the <b>Maximum Closed-Loop Acceleration (Phys. Unit/s<sup>2</sup>)</b> parameter (ID 0x4A).

**ACC? (Get Closed-Loop Acceleration)**

Description:	Queries the acceleration value set with ACC (p. 85).  If all arguments are left out, gets the value of all axes set with ACC.
Format:	ACC? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller
Response:	{<AxisID>="<float> LF}  where

<float> is the acceleration value set with ACC, in physical units/s<sup>2</sup>.

### ADD (Add and Save to Variable)

**Description:** Adds two values and saves the result to a variable (p. 78).

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

**Notes:** 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 only intended for PI service personnel. Users cannot change to a level > 1. Contact the customer service department (p. 191) if you have problems with the parameters for command level 2 or higher.

**Response:** none

**Troubleshooting:** Invalid password

**Notes:** With the C-877, the command levels only determine the write permission for the parameters. The availability of the commands of the C-877 is independent of the active command level.

HPA? (p. 106) lists the parameters including the information on which command level allows write access to them. For further information on using parameters, see "Adapting Settings" (p. 163).

After controller switch-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:	<p>&lt;Level&gt; should be 0 or 1.</p> <p>&lt;Level&gt; = 0 is the default setting, write access is specified for level 0 parameters, read access is specified for all parameters</p> <p>&lt;Level&gt; = 1 allows write access for level 1 parameters (parameters from level 0 are included).</p>

**CPY (Copy Into Variable)**

Description:	<p>Copies a command response to a variable (p. 78).</p> <p>The variable is present in volatile memory (RAM) only.</p>
Format:	CPY <Variable> <CMD?>
Arguments:	<p>&lt;Variable&gt; is the name of the variable to which the command response is to be copied.</p> <p>&lt;CMD?&gt; is one query command in its usual notation. The response has to be a single value and not more.</p>
Response:	None

**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:	<p>{&lt;AxisID&gt;="&lt;string&gt; LF}</p> <p>where</p> <p>&lt;string&gt; is the name of the positioner type assigned to the axis.</p>

Notes: The positioner name is read from the **Stage Name** parameter (ID 0x3C). If the parameter has the value NOSTAGE, the axis is "deactivated". 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`.

You can set the value of the 0x3C parameter specifically to the name of your positioner with SPA (p. 126) or SEP (p. 122). Because the PC software from PI uses the parameter value to configure the C-877 for the connected positioner (p. 51), it is not recommended to change manually with SPA or SEP.

#### 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

#### DEC (Set Closed-Loop Deceleration)

Description: Sets deceleration of specified axes.

DEC can be changed while the axis is in motion.

Format: DEC {<AxisID> <Deceleration>}

Arguments: <AxisID> is one axis of the controller.

<Deceleration> is the deceleration value in physical units/s<sup>2</sup>.

Response: None

Troubleshooting: Illegal axis identifiers

Notes: The DEC setting only takes effect when the specified axis is in closed-loop operation (servo mode ON).

The lowest possible value for <Deceleration> is 0.



DEC changes the value of the **Closed Loop Deceleration (Phys. Unit/s<sup>2</sup>)** parameter (ID 0xC) in the volatile memory of the C-877. The parameter value can be stored as default with WPA (p. 138), for details see "Adapting Settings" (p. 163).

The maximum value that can be set with the DEC command is specified by the **Maximum Closed-Loop Deceleration (Phys. Unit/s<sup>2</sup>)** parameter (ID 0x4B).

### DEC? (Get Closed-Loop Deceleration)

**Description:** Queries the deceleration value set with DEC (p. 90).

If no arguments are specified, queries the value of all axes set with DEC.

**Format:** DEC? [{<AxisID>}]

**Arguments:** <AxisID> is one axis of the controller.

**Response:** {<AxisID>="<float> LF}

where

<float> is the deceleration value set with DEC, in physical units/s<sup>2</sup>.

### DEL (Delay the Command Interpreter)

**Description:** Delays <uint> milliseconds.

**Format:** DEL <uint>

**Arguments:** <uint> is the delay value in milliseconds.

**Response:** None

**Notes:** DEL can only be used in macros. Do not mistake MAC DEL (deletes macros) for DEL (delays). Further information can be found in the description of the MAC command (p. 108) and in the "Controller Macros" (p. 64) section.

### DFH (Define Home Position)

**Description:** Redefines the zero position of the specified axis by setting the position value to zero at the current position.

	If no arguments are specified, DFH defines the zero position of all axes.
Format:	DFH [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller.
Response:	none
Troubleshooting:	Illegal axis identifier
Notes:	DFH sets the current position of the axis to zero and saves the position value which was valid when the command was called as offset in the volatile memory. By adding this offset to the response, the output values of the following commands are adapted to the new zero position:

- POS? (p. 117) (Get the current position)
- TMN? (p. 132) (Get the minimum commandable position)
- TMX? (p. 133) (Get the maximum commandable position)

DFH does **not** change the values of the parameters for the definition of travel range and soft limits (p. 30).

The offset is reset to zero in the following cases:

- When switching on and rebooting the C-877: For all axes
- During referencing: For the affected axis

Example:

```
Send: MOV 1 9.87
Send: POS? 1
Receive: 1=9.8700005
Send: DFH? 1
Receive: 1=0.0000000
Send: TMN? 1
Receive: 1=0.0000000
Send: TMX? 1
Receive: 1=14.9999982
```

Note: Axis 1 is moved to absolute position 9.87 mm. Finally, the current axis position (with POS?), the current offset value (with DFH?), and the minimum and maximum commandable position (with TMN? and TMX?) are queried.

```
Send: DFH 1
Send: POS? 1
Receive: 1=0.0000000
```

```

Send: DFH? 1
Receive: 1=9.8700005
Send: TMN? 1
Receive: 1=-9.8700005
Send: TMX? 1
Receive: 1=5.1299978

```

Note: The axis has not moved. The current axis position was defined as new zero position using DFH. Therefore, the offset value of axis 1 is 9.87 mm. The values for the minimum and maximum commandable position were adapted to the new zero position by adding the offset.

### DFH? (Get Home Position Definition)

Description:	<p>Queries the position value that is currently used as the offset for the specified axis to move the zero position.</p> <p>If no arguments are specified, queries the position value of all axes.</p>
Format:	DFH? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller
Response:	<p>{&lt;AxisID&gt;="&lt;PositionOffset&gt; LF}</p> <p>where</p> <p>&lt;PositionOffset&gt; is the axis position that was valid at the time the last DFH command was processed. This position value is used internally as offset for the calculation of the current axis position.</p>
Troubleshooting:	Illegal axis identifier
Notes:	<p>The axis position that was valid when the last DFH command was processed, is available in the volatile memory as an offset. The offset is reset to zero in the following cases:</p> <ul style="list-style-type: none"> <li>▪ When switching on and rebooting the C-877: For all axes</li> <li>▪ During referencing: For the affected axis</li> </ul> <p>See DFH for an example.</p>

**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 C-877 has 4 data recorder tables with 1024 points per table.

With HDR? (p. 104), you will obtain a list of all available record and trigger options and additional information on the data recording. The number of available data recorder tables can be read with TNR? (p. 133).

For further information, see "Data Recorder" (p. 63).

Recording options available with the corresponding data sources:

- 0=Nothing is recorded
- Data source is the axis:
- 1=Commanded position of axis
  - 2=Actual position of the axis
  - 3=Position error of axis
  - 70=Commanded velocity of axis
  - 71=Commanded acceleration of axis
  - 73=Motor output of axis
  - 74=Kp of axis
  - 75=Ki of axis
  - 76=Kd of axis
  - 80=Signal status register of axis
  - 90=Active parameter set
  - 91=Actual frequency

**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:	<p>The current DRC settings:</p> <p>{&lt;RecTableID&gt;="&lt;Source&gt; &lt;RecOption&gt; LF}</p> <p>where</p> <p>&lt;Source&gt;: is the data source, for example, an axis or a channel of the controller. The source type depends on the record option.</p> <p>&lt;RecOption&gt;: is the type of data to be recorded (record option).</p> <p>The available record options can be queried with HDR? (p. 104).</p>

**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:	<p>{&lt;RecTableID&gt;="&lt;uint&gt; LF}</p> <p>where</p> <p>&lt;uint&gt; specifies the number of points recorded with the last recording</p>

**DRR? (Get Recorded Data Values)**

Description:	<p>Queries the last recorded data.</p> <p>Querying can take some time depending on the number of points to be read!</p>
--------------	-------------------------------------------------------------------------------------------------------------------------

	It is possible to read the data while recording is still in progress.
Format:	DRR? [<StartPoint> <NumberOfPoints> [{<RecTableID>}]]
Arguments:	<p>&lt;StartPoint&gt; is the first point to be read from the data recorder table, starts with index 1.</p> <p>&lt;NumberOfPoints&gt; is the number of points to be read per table.</p> <p>&lt;RecTableID&gt; is one data recorder table of the controller.</p>
Response:	For the recorded data in GCS array format, refer to the separate manual for the GCS array, SM146E, and the example below.
Notes:	<p>If &lt;RecTableID&gt; is not specified, the data is read from all tables with a record option not equal to zero.</p> <p>With HDR? (p. 104), you will obtain a list of all available recording and triggering options as well as additional information on data recording.</p>
Example:	<p>Refer to the description of the DRC command (p. 93) as well as "Data Recorder" (p. 63) for further information.</p> <pre> rtr? 10 drr? 1 20 # REM C-877 # # VERSION = 1 # TYPE = 1 # SEPARATOR = 32 # DIM = 2 # SAMPLE_TIME = 0.0001000 # NDATA = 20 # # NAME0 = Actual Position of Axis AXIS:1 # NAME1 = Position Error of Axis  AXIS:1 # # END_HEADER 5.00000 0.00000 4.99998 0.00002 5.00000 0.00000 5.00000 0.00000 5.00000 0.00000 5.00000 0.00000 </pre>

5.00000	0.00000
4.99998	0.00002
5.00000	0.00000
4.99998	0.00002
5.00000	0.00000
5.00000	0.00000
5.00000	0.00000
5.00000	0.00000
4.99998	0.00002
5.00000	0.00000
4.99998	0.00002
4.99998	0.00002
5.00000	0.00002
4.99998	0.00004

### 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:** At present, only 0 is valid for <RecTableID>; this means that the given trigger source is set for all data recorder tables that have a record option that is not zero.

Irrespective of the trigger option set, the data recording is always triggered when a step response measurement is carried out with STE (p. 130).

With HDR? (p. 104), you will obtain a list of all available record and trigger options and additional information on the data recording.

For further information, see the description of the DRC command (p. 93) as well as "Data Recorder" (p. 63).

Available trigger options:	0 = default setting Data recording is triggered by STE; <Value> must be a dummy.
	1 = any command changing target position e.g., MVR (p. 115), MOV (p. 114); <Value> must be a dummy.
	2 = next command resets trigger after execution; <Value> must be a dummy.
	6 = any command changing target position, reset trigger after execution e.g., MVR, MOV; resets trigger after execution; <Value> must be a dummy.
	7 = SMO command, reset trigger after execution resets trigger after execution; <Value> must be a dummy.

**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. 97).
Notes:	Because all data record tables of the C-877 have the same trigger source, the DRT? response is specified as a single line of the form  0=<TriggerSource> <Value>



**ERR? (Get Error Number)**

Description:	<p>Get error code &lt;int&gt; of the last occurred error and reset the error to 0.</p> <p>Only the last error is buffered. You should therefore call ERR? after each command.</p> <p>The error codes and their descriptions are listed in "Error Codes" (p. 139).</p>
Format:	ERR?
Arguments:	None
Response:	The error code of the last error that occurred (integer).
Troubleshooting:	Communication breakdown
Notes:	<p>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.</p> <ul style="list-style-type: none"> <li>➤ If possible, access the controller with one instance only.</li> <li>➤ 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).</li> </ul> <p>If the cause of an error continues, the corresponding error code is immediately set again after a query with ERR?.</p>

**FED (Find Edge)**

Description:	<p>Moves the specified axis to a specified signal edge.</p> <p>FED does not set a certain position value at the selected edge (in contrast to the FNL (p. 100), FPL (p. 102), and FRF (p. 102) commands for referencing), i.e., the axis is not "referenced" after using FED.</p> <p>If multiple axes are specified in the command, they are moved synchronously.</p>
Format:	FED {<AxisID> <EdgeID> <Param>}
Arguments:	<p>&lt;AxisID&gt; is one axis of the controller.</p> <p>&lt;EdgeID&gt; is the type of edge the axis has to move to. See</p>

below for available edge types.

<Param> depends on the selected edge and qualifies it.  
See below for details.

Response: None

Troubleshooting: Illegal axis identifier; limit switches and/or reference switch are disabled (see below); SVO? (p. 132) responds with the value 0.

Notes: Servo mode must be switched on with SVO (p. 131) for the commanded axis prior to using this command (closed-loop operation).

The firmware of the C-877 determines the following based on parameters:

- Is a reference switch present (parameter 0x14)?
- Are limit switches present (parameter 0x32)?
- If the reference switch is represented by an index pulse: How should the move to the index pulse take place (parameters 0x70, 0x78, 0x79)?

According to the values of those parameters, the C-877 activates or deactivates FED motions to the corresponding signal edges. Adapt the parameter values to your hardware using SPA (p. 126) or SEP (p. 122). For further information, see "Adapting Settings".

FED can be used to measure the physical travel range of a new mechanical system and therefore determine the values for the corresponding parameters:

- Distance from the negative to the positive limit switch
- Distance between the negative limit switch and the reference switch (parameter ID 0x17)
- Distance between the reference switch and the positive limit switch (parameter ID 0x2F).

For further information, see "Travel Range and Soft Limits" (p. 30).

The motion can be stopped by #24 (p. 85), STP (p. 130), and HLT (p. 105).

Available edge types and parameters:

The following edge types with their parameter settings are available:

- 1 = negative limit switch, <Param> must be 0
- 2 = positive limit switch, <Param> must be 0
- 3 = reference switch, <Param> must be 0

**FNL (Fast Reference Move To Negative Limit)**

**Description:** Performs a reference move.

Moves the specified axis to the negative physical limit of its travel range and sets the current position to a defined value. See below for details.

If multiple axes are specified in the command, they are moved synchronously.

**Format:** FNL [{<AxisID>}]

**Arguments:** <AxisID> is a controller's axis, all axes are affected if not specified.

**Response:** None

**Troubleshooting:** Illegal axis identifier

**Notes:** Servo mode must be switched on with SVO (p. 131) for the commanded axis prior to using this command (closed-loop operation).

If the reference move was successful, absolute motion will then be possible in closed-loop operation.

The negative physical limit of the travel range is represented by the negative limit switch of the positioner. The difference in the values of the parameters 0x16 and 0x17 is set as the current position when the axis is at the negative limit switch.

The motion can be stopped by #24 (p. 85), STP (p. 130) and HLT (p. 105).

Use FRF? (p. 103) to check whether the reference move was successful.

For best repeatability, referencing must always be done in the same way.

If soft limits (parameters 0x15 and 0x30) are used to reduce the travel range, the limit switches cannot be used for reference moves.

For further information, see "Referencing" (p. 34) and "Travel Range and Soft Limits" (p. 30).

**FPL (Fast Reference Move To Positive Limit)**

Description:	<p>Starts a reference move.</p> <p>Moves the specified axis to the positive physical limit of its travel range and sets the current position to a defined value. See below for details.</p> <p>If multiple axes are specified in the command, they are moved synchronously.</p>
Format:	FPL [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller, if not specified, all axes are involved.
Response:	none
Troubleshooting:	Illegal axis identifier
Notes:	<p>Servo mode must be switched on with SVO (p. 131) for the commanded axis prior to using this command (closed-loop operation).</p> <p>If the reference move was successful, absolute motion will then be possible in closed-loop operation.</p> <p>The positive physical limit of the travel range is represented by the positive limit switch of the positioner. The sum of the values of the parameters 0x16 and 0x2F is set as the current position when the axis is at the positive limit switch.</p> <p>The motion can be stopped by #24 (p. 85), STP (p. 130) and HLT (p. 105).</p> <p>Use FRF? (p. 103) to check whether the reference move was successful.</p> <p>For best repeatability, referencing must always be done in the same way.</p> <p>If soft limits (parameters 0x15 and 0x30) are used to reduce the travel range, the limit switches cannot be used for reference moves.</p> <p>For further information, see "Referencing" (p. 34) and "Travel Range and Soft Limits" (p. 30).</p>

**FRF (Fast Reference Move To Reference Switch)**

Description:	Starts a referencing 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 a controller's axis, all axes are affected if not specified.
Response:	None
Troubleshooting:	Illegal axis identifier
Notes:	<p>Servo mode must be switched on with SVO (p. 131) for the commanded axis prior to using this command (closed-loop operation).</p> <p>If the reference move was successful, absolute motion will then be possible in closed-loop operation.</p> <p>The value of the parameter 0x16 is set as the current position when the axis is at the reference switch.</p> <p>The motion can be stopped by #24 (p. 85), STP (p. 130) and HLT (p. 105).</p> <p>Use FRF? (p. 103) to check whether the reference move was successful.</p> <p>Use FNL (p. 100) or FPL (p. 102) instead of FRF to do a reference move for an axis which has no reference switch.</p> <p>For best repeatability, referencing must always be done in the same way.</p> <p>For further information, see "Referencing" (p. 34) and "Travel Range and Soft Limits" (p. 30).</p>

**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
Notes:	An axis is considered as "referenced" when the current position value is set to a known position. This is the case when a reference move was successfully done with FNL (p. 100), FPL (p. 102) or FRF (p. 102) or when the position was set directly with POS (p. 116) (depending on the referencing method selected with RON (p. 118)).

#### GOH (Go To Home Position)

Description:	Moves the specified axis to the zero position.
	GOH [{<AxisID>}] is the same as MOV {<AxisID> 0}
	The motion can be stopped by #24 (p. 85), STP (p. 130), and HLT (p. 105).
Format:	GOH [{<AxisID>}]
Arguments:	<AxisID>: Is one axis of the controller; if not specified, all axes are affected.
Response:	None
Troubleshooting:	Illegal axis identifier
Notes:	Servo mode must be switched on for the commanded axis prior to using this command (closed-loop operation).

#### 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
```

#### **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

#### **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. 85) and STP (p. 130) 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:	<p>HLT stops motion with specified system deceleration regarding system inertia. Does not apply to trajectories.</p> <p>HLT stops all motion caused by motion commands (e.g., MOV (p. 114), MVR (p. 115), GOH (p. 104), STE (p. 130), SMO (p. 124)), the command for referencing (FRF (p. 102)), and macros (MAC (p. 108)).</p> <p>After the axis has been stopped, its target position is set to its current position.</p>

#### 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. 171) for further information.
Format:	HPA?
Arguments:	None
Response	<p>{&lt;PamID&gt;="&lt;string&gt; LF}</p> <p>where</p> <p>&lt;PamID&gt; is the ID of one parameter, hexadecimal format</p> <p>&lt;string&gt; is a string which describes the corresponding parameter.</p> <p>The string has following format:</p> <p>&lt;CmdLevel&gt;TAB&lt;MaxItem&gt;TAB&lt;DataType&gt;TAB&lt;FunctionGroupDescription&gt;TAB&lt;ParameterDescription&gt;[{TAB&lt;PossibleValue&gt;="&lt;ValueDescription&gt;}]</p> <p>where</p> <p>&lt;CmdLevel&gt; is the command level which allows write access to the parameter value.</p> <p>&lt;MaxItem&gt; is the maximum number of items of the same</p>



type which are affected by the parameter. With the C-877, an "item" is an axis 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 to which the parameter belongs.

<ParameterDescription> is the parameter name.

<PossibleValue> is one value from the allowed data range.

<ValueDescription> is the meaning of the corresponding value.

The parameters listed with HPA? can be changed and/or saved using the following commands:

SPA (p. 126) influences the parameter settings in volatile memory (RAM).

WPA (p. 138) copies parameter settings from volatile to nonvolatile memory.

SEP (p. 122) writes parameter settings directly into nonvolatile memory (without changing settings in volatile memory).

RPA (p. 119) resets volatile memory to the values from nonvolatile memory.

### 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 that the macro execution pointer jumps back to the previous line; 0 means that the command is executed again, which is the same behavior as with WAC (p. 138). 1 jumps to the next line, making the command unnecessary, and 2 jumps over the next command. Jumps are only permitted in the current macro.
	<CMD?> is one query command in its usual notation. The response must 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 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

### 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

Notes: The C-877 firmware detects the presence or absence of limit switches using a parameter (ID 0x32). Corresponding to the value of this parameter, the C-877 enables or disables stopping of motion at the limit switches and reference moves using FNL (p. 100) or FPL (p. 102).

Adapt the parameter value to your hardware using SPA (p. 126) or SEP (p. 122). For further information, see "Limit Switch Detection" (p. 30).

### 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 <macro name>  
 MAC DEF <macro name>  
 MAC DEF?  
 MAC DEL <macro name>  
 MAC END  
 MAC ERR?  
 MAC FREE?  
 MAC NSTART <macro name> <uint> [<String1> [<String2>]]  
 MAC START <macro name> [<String1> [<String2>]]

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 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 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 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 <macro name> <uint> [<String1> [<String2>]]

Repeats the specified macro <uint> times. The macro is re-run each time.

<String1> and <String2> are optional arguments which specify the values for local variables 1 and 2 used in the specified macro. <String1> and <String2> can be specified directly or via the values of variables. Macro will not run if the macro contains local variables but <String1> and <String2> are not specified in the MAC NSTART command. Refer to "Variables" (p. 78) for further details.

MAC START <macroname> [<String1> [<String2>]]

Runs the specified macro once. <String1> and <String2> have 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: Running a macro is not allowed when a macro is being recorded.

When a macro is recorded for a controller whose address is different from 1, the target address must be part of each command line, but will not become part of the macro content. PIMikroMove automatically sends the target address during the macro recording so that it does not have to be entered there. You will find further information in "Working with Macros" (p. 66) and "Target and Sender Address" (p. 77).

The `MAC BEG` and `MAC END` commands may not be specified when macros are recorded in the **Controller macros** tab in PIMikroMove.

A macro can be overwritten by a macro with the same name.

Macros can contain local and global variables. Refer to "Variables" (p. 78) for further information.

A running macro sends no responses to any interface.

Depending on the value of parameter 0x72 (**Ignore Macro Error?**), the following options exist when an error is caused by a running macro:

0 = Macro running is aborted (default).

1 = The error is ignored and the macro continues to run.

MAC ERR? always reports the last error that occurred while the a macro was running irrespective of the parameter setting.

The following commands provided by the C-877 can only be used in macros:

DEL (p. 91), JRC (p. 107), MEX (p. 113) and WAC (p. 138).

A macro can start another macro. The maximum number of nesting levels is 5. A macro can call itself to form an infinite loop.

All commands can be sent from the command line while a macro is running. The macro content and motion commands received from the command line can overwrite each other.

Macro execution can be stopped with #24 (p. 85) and STP (p. 130).

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

A macro cannot be deleted while it is running.

You can query with #8 (p. 84) if a macro is currently running on the controller.

**Note: The number of write cycles in the nonvolatile memory is limited. Only record macros if this is necessary.**

#### 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.

### MAT (Calculate And Save To Variable)

Description: Carries out a mathematical operation or bit operation and saves the result as a variable (p. 78).

The variable is in volatile memory (RAM) only.

Format: MAT <Variable> "=" <FLOAT1> <OP> <FLOAT2>

Arguments: <Variable> is the name of the variable where the result is to be saved.

<FLOAT1> and <FLOAT2> are the values for calculating the result. They can be specified directly or via the value of a variable.

<OP> is the operator to be used: The following operators are possible:

<OP>	Operation	Type
+	Addition	Mathematical operation
-	Subtraction	Mathematical operation
*	Multiplication	Mathematical operation
AND	UND	Bit operation
OR	ODER	Bit operation
XOR	XOR	Bit operation

Important: There must be a blank space before and after each "=" and the operator!

Response: None

Notes: Using MAT to set local variables is only possible in macros.

- Example 1: Send: `MAT TARGET = ${POS} * 2.0`  
The `TARGET` variable contains 2.0 times the value of the `POS` variable.
- Example 2: Send: `MAT TARGET = 2 * 0x10`  
Send: `VAR? TARGET`  
Receive: `TARGET=32`  
NOTICE: The values from which the result is to be calculated can be written in hexadecimal or decimal format. The result is always output in decimal format.
- Example 3: Send: `MAT INVERT = 0x45 XOR 0xFF`  
Send: `VAR? INVERT`  
Receive: `INVERT=186`  
NOTICE: The bit operation XOR with the value 0xFF corresponds to an inversion of the value 0x45. The result is output in decimal format.

**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. 138).

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:	<p>&lt;AxisID&gt; is one axis of the controller.</p> <p>&lt;Position&gt; is the absolute target position in physical units.</p>
Response:	none
Notes:	<p>The servo mode must be switched on when this command is used (closed-loop operation).</p> <p>The target position must be within the soft limits. Use TMN? (p. 132) and TMX? (p. 133) to query the current valid soft limits.</p> <p>The motion can be stopped by #24 (p. 85), STP (p. 130), and HLT (p. 105).</p> <p>During a motion, a new motion command resets the target to a new value and the old value may never be reached. This is also valid with macros: Motion 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.</p>

Example 1: Send: `MOV 1 10`  
 Note: Axis 1 moves to 10 (target position in mm)

Example 2: Send: `MOV 1 243`  
 Send: `ERR?`  
 Receive: `7`  
 Note: The axis does not move. The error code "7" in the reply to the ERR? command (p. 98) indicates that the target position specified 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

### 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

Notes: The servo mode must be switched on when this command is used (closed-loop operation).

The target position must be within the soft limits. Use TMN? (p. 132) and TMX? (p. 133) to get the currently valid soft limits, and MOV? (p. 114) to get the current target.

The motion can be stopped by #24 (p. 85), STP (p. 130), and HLT (p. 105).

During a motion, a new motion command resets the target to a new value and the old value may never be reached. This is also valid with macros: Motion 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.

Example: Send: MOV 1 0.5  
Note: This is an absolute motion.

Send: POS? 1

Receive: 1=0.500000

Send: MOV? 1

Receive: 1=0.500000

Send: MVR 1 2

Note: This is a relative motion.

Send: POS? 1

Receive: 1=2.500000

Send: MVR 1 2000

Note: New target position of axis 1 would exceed motion range. Command is ignored, i.e., the target position remains unchanged, and the axis does not move.

Send: MOV? 1

Receive: 1=2.500000

Send: POS? 1

Receive: 1=2.500000

### 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: The detection of the on-target state is only possible in closed-loop operation (servo mode ON).

The on-target state is influenced by the settings for the settling window (parameter 0x406 and 0x407) and the delay time (parameter 0x3F). For details, see "On-Target State" (p. 27).

### POS (Set Real Position)

Description: Sets the current position of the axis (does not cause motion).

Format: POS {<AxisID> <Position>}

Arguments:	<p>&lt;AxisID&gt; is one axis of the controller.</p> <p>&lt;Position&gt; is the new current position in physical units.</p>
Response:	None
Troubleshooting:	Illegal axis identifier
Notes:	<p>It is only possible to set the current position with POS when the referencing method "0" is selected, see RON (p. 118).</p> <p>An axis is considered to be "referenced" when the position has been set with POS (for more information, see "Referencing" (p. 34)).</p> <p>The minimum and maximum commandable positions (TMN? (p. 132), TMX? (p. 133)) are not adapted when a position is set with POS. This can result in target positions which are allowed by the C-877 but cannot be reached by the hardware. Target positions are also possible that can be reached by the hardware but are refused by the C-877. Furthermore, the zero position can be outside of the physical travel range after using POS.</p>

**POS? (Get Real Position)**

Description:	<p>Queries the current axis position.</p> <p>If no arguments are specified, the current position of all axes is queried.</p>
Format:	POS? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller.
Response:	<p>{&lt;AxisID&gt;="&lt;float&gt; LF}</p> <p>where</p> <p>&lt;float&gt; is the current axis position in physical units.</p>
Troubleshooting:	Illegal axis identifier

**RBT (Reboot System)**

Description:	Reboots system. The controller behaves the same as after switching on.
Format:	RBT
Arguments:	none
Response:	none
Notes:	RBT cannot be used in macros. This is to avoid problems with startup macro execution.

**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 (Set Reference Mode)**

Description:	Selects the referencing method for the specified axes
Format:	RON {<AxisID> <ReferenceOn>}
Arguments:	<AxisID> is one axis of the controller.  <ReferenceOn> is the referencing method. Can be 0 or 1. 1 is default. See below for details.
Response:	None
Troubleshooting:	Illegal axis identifier
Notes:	<ReferenceOn> = 0: An absolute position value can be assigned with POS (p. 116) or a reference move can be started with FRF (p. 102), FNL (p. 100) or FPL (p. 102). Relative motion with MVR is possible, even when

referencing has not been done for the axis.

<ReferenceOn> = 1: A reference move for the axis must be started with FRF, FNL or FPL. Using POS is not allowed. Motion in closed-loop operation is only possible when the axis has been referenced.

For further information, see "Referencing" (p. 34) and "Travel Range and Soft Limits" (p. 30).

### **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: Further information can be found in the description of the RON command (p. 118).

### **RPA (Reset Volatile Memory Parameters)**

Description: Resets the specified parameter of the specified element. The value from nonvolatile memory is written into volatile memory.

Related commands:

With HPA? (p. 106) you can obtain a list of the available parameters. SPA (p. 126) influences the parameter settings in volatile memory, WPA (p. 138) writes parameter settings from volatile to nonvolatile memory, and SEP (p. 122) writes parameter settings directly into nonvolatile memory (without changing the settings in volatile memory).

See SPA for an example.

Format:	RPA [{<ItemID> <PamID>}]
Arguments:	<p>&lt;ItemID&gt; is the element for resetting a parameter. See below for details.</p> <p>&lt;PamID&gt; is the parameter ID, can be written in hexadecimal or decimal format. See below for details.</p>
Response:	none
Troubleshooting:	Illegal element identifier, wrong parameter ID
Notes:	<p>The information from the positioner's ID chip and the positioner databases are loaded into the volatile memory of the C-877. The loaded data is overwritten by RPA. Only use RPA if you are sure that the C-877 functions correctly with the parameter values from the nonvolatile memory.</p> <p>With the C-877, you can reset either all parameters or specifically, one single parameter with RPA.</p>
Available item IDs and parameter IDs:	<p>An item is an axis; the identifier can be changed with SAI (p. 121). For further information, see "Commandable Items" (p. 13).</p> <p>Valid parameter IDs are specified in "Parameter Overview" (p. 171).</p>

**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:	<p>The duration of the recording can be calculated as follows:</p> $\text{Rec. duration} = \text{cycle time of the servo loop} * \text{RTR value} * \text{number of points}$ <p>where</p> <p>the cycle time of the servo loop for the C-877 is 100 µs</p>

the number of points for the C-877 is 1024 (length of data recorder table)

For further information, see "Data Recorder" (p. 63).

The record table rate set with RTR is saved in volatile memory (RAM) only.

#### **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).

#### **SAI (Set Current Axis Identifiers)**

Description:	Sets the axis identifiers for the specified axes.  After it was set with SAI, the new axis identifier must be used as <AxisID> in all axis-related commands.
Format:	SAI {<AxisID> <NewIdentifier>}
Arguments:	<AxisID> is one axis of the controller  <NewIdentifier> is the new identifier to use for the axis, see below for details
Response:	none
Notes:	An axis could be identified with up to 8 characters. Use TVI? (p. 134) to ask for valid characters. The new axis identifier is saved automatically and is therefore still available after rebooting or switching on the next time.

**SAI? (Get List Of Current Axis Identifiers)**

Description:	Queries the axis identifiers.
	Refer also to "Commandable Elements" (p. 13).
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.
Notes:	If the <b>Stage Name</b> parameter (0x3C) has the value NOSTAGE, the axis is "deactivated". A deactivated axis is not accessible for axis-related commands (e.g. motion commands or position queries) and is only included in the response to <code>SAI? ALL</code> .

**SEP (Set Non-Volatile Memory Parameters)**

Description:	Sets a parameter of a specified element to a different value in nonvolatile memory, where it becomes the new default.
	After parameters were set with SEP, you can use RPA (p. 119) to activate them (write them to volatile memory) without controller reboot.
	<b>Note that this command is for setting hardware-specific parameters. Wrong values may lead to improper operation or damage of your hardware!</b>
	Related commands:
	HPA? (p. 106) returns a list of the available parameters.
	SPA (p. 126) writes parameter settings into volatile memory (without changing the settings in nonvolatile memory).
	WPA (p. 138) writes parameter settings from volatile to nonvolatile memory.
Format:	SEP <Pswd> {<ItemID> <PamID> <PamValue>}



Arguments	<p>&lt;Pswd&gt; is the password for writing to the nonvolatile memory; the default value is "100".</p> <p>&lt;ItemID&gt; is the element for changing a parameter in the nonvolatile memory. See below for details.</p> <p>&lt;PamID&gt; is the parameter identifier, can be written in hexadecimal or decimal format. See below for details.</p> <p>&lt;PamValue&gt; is the value for setting the specified parameter of the specified element.</p>
Response:	None
Troubleshooting:	Illegal element identifier, wrong parameter ID, invalid password
Notes:	<p><b>Note that the number of write cycles in the nonvolatile memory is limited. Write default settings only if necessary.</b></p> <p>With the C-877 you can only write one parameter per SEP command.</p>
Available item IDs and parameter IDs:	<p>An element is an axis (the identifier can be changed with SAI (p. 121)) or the entire system. Refer to "Commandable Elements" (p. 13) for further information.</p> <p>Valid parameter IDs are specified in "Parameter Overview" (p. 171).</p>

### SEP? (Get Nonvolatile Memory Parameters)

Description:	<p>Queries the value of a parameter of a specified element from nonvolatile memory.</p> <p>With HPA? (p. 106) you can obtain a list of the available parameters and their IDs.</p>
Format:	SEP? [{<ItemID> <PamID>}]
Arguments:	<p>&lt;ItemID&gt; is the element for querying a parameter value from nonvolatile memory. See below for details.</p> <p>&lt;PamID&gt; is the parameter ID, can be written in hexadecimal or decimal format. See below for details.</p>

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
Notes:	With the C-877, you can query either all parameters or specific individual parameters with each SEP? command.
Available item IDs and parameter IDs:	An item is an axis (the identifier can be changed with SAI (p. 121)) or the entire system. For further information, see "Commandable Items" (p. 13).
	Valid parameter IDs are given in "Parameter Overview" (p. 171).

### SMO (Set Open-Loop Control Value)

Description:	Sets control value directly to move the axis. Profile generator (if present), sensor feedback and servo algorithm are not taken into account.
	Servo mode must be switched off when using this command (open-loop operation).
Format:	SMO {<AxisID> <ControlValue>}
Arguments	<AxisID> is one axis of the controller.
	<ControlValue> is the new control value (dimensionless). See below for details.
Response:	None
Troubleshooting:	Illegal axis identifier
Notes:	Servo mode is switched on for one of the specified axes.
	<b>NOTICE: In the case of large control values, the positioner can strike the hard stop despite the limit switch function. This can cause damage to equipment.</b>
	The unsigned control value may not be greater than the value of the <b>Maximum Motor Output</b> parameter (0x9).

When this parameter is set to its maximum (32767), <ControlValue> ranges from -32766 to 32766 (dimensionless). <ControlValue> controls the piezo voltage for the axis. The sign of the value determines the direction of motion: -32766 corresponds to the maximum amplitude of the piezo voltage in the negative direction of motion and 32766 corresponds to the maximum amplitude of the piezo voltage in the positive direction of motion. For further information, see "Supported Motor Types" (p. 28).

When a high control value remains set over a long period of time, the connected positioner can heat up. Overheating can result in damage to the positioner.

The **PID Maximum Output Time (s)** parameter (ID 0x7B) specifies the maximum time period for which a high control value may be set in closed-loop operation. A high control value is present when the following applies:  
Current absolute measure of the control value  $\geq 95\%$  of **Maximum Motor Output** (ID 0x9). For further information, see "Protection Against Overheating" (p. 61).

The **Range Limit Min** (0x07000000) and **Range Limit Max** (0x07000001) parameters can be used as soft limits for motions in open-loop operation with SMO: When the current position reaches these values, the control value is set to zero and the motion is stopped. The axis can be moved again as soon as the value for the soft limit has been increased or decreased.

Example:

Send: SMO 1 -16000

Note: The control value is about half the maximum control value. The axis moves in negative direction.

### SMO? (Get Control Value)

Description: Gets last valid control value of given axis.

Format: SMO? [{<AxisID>}]

Arguments <AxisID> is one axis of the controller

Response: {<AxisID>="<float> LF}

where

<float> is the last valid control value (dimensionless). For details see below.

Troubleshooting: Illegal axis identifier

Notes: The control value which is returned by SMO? can be the result of the servo algorithm and other corrections, or it can be the value set by an SMO command in open-loop operation.

For further information, see SMO (p. 124) and the block diagram (p. 13).

### 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

Parameter changes are also lost when the parameters are reset to their default values with RPA (p. 119).

**Note that this command is for setting hardware-specific parameters. Wrong values may lead to improper operation or damage of your hardware!**

Related commands:

HPA? (p. 106) returns a list of the available parameters.

SEP (p. 122) writes parameter settings directly into nonvolatile memory (without changing the settings in volatile memory).

WPA (p. 138) writes parameter settings from volatile to nonvolatile memory.

RPA resets volatile memory to the value in nonvolatile memory.

Troubleshooting:	Illegal item identifier, wrong parameter ID, value out of range
Notes:	With the C-877, you can write only one parameter per SPA command.
Available item IDs and parameter IDs:	<p>An item is an axis (the identifier can be changed with SAI (p. 121)) or the entire system. For further information, see "Commandable Items" (p. 13).</p> <p>Valid parameter IDs can be found in the parameter overview (p. 171).</p>
Example 1:	<p>Send: SPA 1 0x411 100</p> <p>Note: Sets the P term of the servo algorithm for axis 1 to 100 for parameter group 1; the parameter ID is written in hexadecimal format</p> <p>Send: SPA 1 1041 150</p> <p>Note: Sets the P term of the servo algorithm for axis 1 to 150 for parameter group 1; the parameter ID is written in decimal format</p>
Example 2:	<p>For parameter group 2, the P, I, and D parameters of the servo algorithm must be adapted to a new load that is applied to the connected mechanical system.</p> <p>Send: SPA 1 0x421 150</p> <p>Note: The P term is set to 150 for axis 1. The setting is made in volatile memory only.</p> <p>Now set the I and D terms in volatile memory using SPA and then test the functioning of the system. If the closed-loop system performance proves satisfactory and you want to use this system configuration as default, save the parameter settings from volatile to nonvolatile memory.</p> <p>Send: WPA 100</p> <p>Note: See the command description for WPA (p. 138) for details on the extent of the saved settings.</p>
Example 3:	<p>Send: SEP 100 LEFT 0xA 20</p> <p>Note: The maximum velocity must be set to 20 mm/s for</p>

axis LEFT (axis was renamed with SAI). The setting is made in nonvolatile memory and is therefore the new default, but is not yet active. To use the new settings immediately, you now have to load them to volatile memory (otherwise they would become active after the next power-on or reboot of the controller).

Send: `RPA`

Note: The new configuration is active now.

Send: `SPA? LEFT 0xA`

Receive: `LEFT 0xA=20.00000`

Note: Check the parameter settings in volatile memory.

### 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. 106).
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
Notes:	With the C-877, you can query either all parameters or specific individual parameters for each SPA? command.
Available element IDs and parameter IDs:	An element is an axis (the identifier can be changed with SAI (p. 121)) or the entire system. For further information, see "Commandable Elements" (p. 13).
	Valid parameter IDs can be found in the parameter overview (p. 171).

**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.

Note: This command is identical in function to #4 (p. 83) which should be preferred when the controller is performing time-consuming tasks.

Possible register IDs and response values: <ItemID> is one axis of the controller.

<RegisterID> can be 1.

<Value> is the bit-mapped response and is returned as the sum of the following individual codes in hexadecimal format:

Bit	15	14	13	12	11	10	9	8
Description	On-target state	Determines the reference value	In motion	Servo mode on	-	-	-	Error flag

Bit	7	6	5	4	3	2	1	0
Description	-	-	-	-	Axis is referenced	Positive limit switch	Reference point switch	Negative limit switch

Example: Send: #4  
 Receive: 0x900A  
 = Binary format: **1001000000001010**  
 Note: The response is in hexadecimal format. It means (displayed in bold):  
 The axis is at the target position (On-Target-Status; bit 15),  
 servo mode is switched on (bit 12),  
 no errors occurred (bit 8),  
 the axis has already been referenced (bit 3),  
 the positioner is at the position of the reference switch (bit 1).

### 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. 93).

The recorded data can be read with the DRR? command (p. 95).

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: Servo mode must be switched on for the commanded axis prior to using this command (closed-loop operation).

The target position must be within the soft limits. Use TMN? (p. 132) and TMX? (p. 133) to get the currently valid soft limits, and MOV? (p. 114) to get the current target.

Notes: A "step" consists of a motion with the specified amplitude which is performed relative to the current position.

### 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. 85).
Format:	STP
Arguments:	None
Response:	None
Troubleshooting:	Communication breakdown
Notes:	<p>SPA stops all motion caused by motion commands (e.g., MOV (p. 114), MVR (p. 115), GOH (p. 104), STE (p. 130), SMO (p. 124)), follow trajectory (TGS), the command for referencing (FRF (p. 102)), and macros (MAC (p. 108)). Also stops macro running.</p> <p>After the axis has stopped, its target position is set to its current position.</p> <p>HLT (p. 105) in contrast to STP stops motion with specified system deceleration regarding system inertia. Does not apply to trajectories.</p>

### SVO (Set Servo Mode)

Description:	Sets the servo mode for specified axes (open-loop or closed-loop operation).
Format:	SVO {<AxisID> <ServoState>}
Arguments:	<p>&lt;AxisID&gt; is one axis of the controller</p> <p>&lt;ServoState&gt; can have the following values:  0 = servo mode off (open-loop operation)  1 = servo mode on (closed-loop operation)</p>
Response:	None
Troubleshooting:	Illegal axis identifier
Notes:	<p>When switching from open-loop to closed-loop operation, the target is set to the current position to avoid jumps of the mechanical system.</p> <p>The current state of the servo mode determines the applicable motion commands:  Servo mode on: Use MOV (p. 114), MVR (p. 115) or GOH (p. 104).  Servo mode off: Use SMO (p. 124).</p> <p>The servo mode must be switched on before reference</p>

moves can be started with FRF (p. 102), FNL (p. 100) or FPL (p. 102).

When the servo mode is switched off while the axis is moving, the axis stops.

If a motion error occurs (p. 61), servo mode is switched off.

### **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

### **TCV? (Get Commanded Closed-Loop Velocity)**

**Description:** Queries the current value of the velocity (value calculated by the profile generator).

**Format:** TCV? [{<AxisID>}]

**Arguments:** <AxisID> is one axis of the controller.

**Response:** {<AxisID>=" "<float> LF}

where

<float> is the velocity value in physical units per second.

### **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
Note:	The minimum commandable position is defined by the parameter 0x30. When redefining the zero position with the DFH (p. 91) command, the minimum commandable position is automatically adapted to the new zero position.

#### 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
Note:	The maximum commandable position is defined by the parameter 0x15. When redefining the zero position with the DFH (p. 91) command, the maximum commandable position is automatically adapted to the new zero position.

#### 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 C-877 has four data recorder tables with 1024 data points per table.

For further information, see "Data Recorder" (p. 63).

### 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
Notes:	<p>The C-877 firmware detects the presence or absence of a reference switch via a parameter (ID 0x14). According to the value of this parameter, the C-877 enables or disables reference moves to the reference switch (FRF command (p. 102)).</p> <p>Adapt the parameter value to your hardware using SPA (p. 126) or SEP (p. 122).</p> <p>For further information, see "Reference Switch Detection" (p. 29).</p>

### TVI? (Tell Valid Character Set For Axis Identifiers)

Description:	<p>Returns a string with characters which can be used for axis identifiers.</p> <p>Use SAI (p. 121) to change the axis identifiers and SAI? (p. 121) to ask for the current valid axis identifiers.</p>
Format:	TVI?
Arguments:	None
Response:	<string> is a list of characters
Notes:	<p>With the C-877, the string consists of</p> <p>1234567890ABCDEFGHIJKLMNPOQRSTUVWXYZ_-</p>

**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. 78) 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. 78) 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 braces 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. 87) for another example.

**VAR? (Get Variable Values)**

Description:	Gets values of variables.
	If VAR? is combined with CPY (p. 89), JRC (p. 107), MEX (p. 113) or WAC (p. 138), the response to VAR? has to be a single value and not more.
	Refer to “Variables” (p. 78) 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. 78) 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.
Notes:	Local variables can be queried using VAR? only when a macro with local variables is running. See “Variables” (p. 78) for details regarding local and global variables.
Example:	See ADD (p. 87) for an example.

**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 VEL setting only takes effect when the specified axis is in closed-loop operation (servo mode ON).
	The lowest possible value for <Velocity> is 0.
	The velocity can be changed with VEL while the axis is moving.

VEL changes the value of the **Closed-Loop Velocity (Phys. Unit/s)** parameter (ID 0x49) in the volatile memory of C-877. The parameter value can be stored as default with WPA (p. 138), for details see "Adapting Settings" (p. 163).

The maximum value that can be set with the VEL command is specified by the **Maximum Closed-Loop Velocity (Phys. Unit/s)** parameter, ID 0xA.

### 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.
Notes:	VEL? gets the value of the velocity for closed-loop operation commanded with VEL (value of the <b>Closed Loop Velocity (Phys. Unit/s)</b> parameter (ID 0x49) in the volatile memory).

### VER? (Get Versions Of Firmware And Drivers)

Description:	Gets the versions of the firmware of the C-877 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.

**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. 113).

**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.

**WPA (Save Parameters To Non-Volatile Memory)**

**Description:** Writes the currently valid value of a parameter of a specified element from volatile memory (RAM) to nonvolatile memory. The values saved this way become the default values.

**Note:** If the current parameter values are incorrect, this can cause a fault in the system. Make sure that the



**parameter settings are correct before you execute the WPA command.**

RAM settings not saved with WPA will be lost when the controller is switched off or rebooted or when RPA (p. 119) is used to restore the parameters.

You can obtain a list of all available parameters with HPA? (p. 106).

Use SPA? (p. 126) to check the current parameter settings in volatile memory.

See SPA (p. 126) for an example.

Format: WPA <Pswd> [{<ItemID> <PamID>}]

Arguments: <Pswd> is the password for writing to the nonvolatile memory. See below for details.

<ItemID> 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 item identifier, wrong parameter ID, invalid password

**Note that the number of write cycles in the nonvolatile memory is limited. Write default settings only if necessary.**

Notes: Parameter values can be changed in the volatile memory with SPA (p. 126).

Valid passwords: The password for writing to the nonvolatile memory is "100".

Element and parameter IDs: It is not possible to specifically select individual items and parameters for saving with the C-877; i. e., <ItemID> and <PamID> are ignored.

## 8.7 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_ESTIMATE	While MOV! is running position can only be

	ON	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 occurred during macro execution
1013	PI_CNTR_NO_MACRO_OR_EMPTY	No macro with given name on controller, or macro is empty

1015	PI_CNTR_INVALID_ARGUMENT	One or more arguments given to function is invalid (empty string, index out of range, ...)
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
1028	PI_CNTR_UNKNOWN_CHANNEL_IDENTIFIER	Unknown channel identifier
1063	PI_CNTR_EXT_PROFILE_UNALLOWED_COMMAND	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
2100	PI_CNTR_FEATURE_LICENSE_INVALID	Entered license is invalid
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	Given 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	Given 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 given 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 given 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

	E	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
-64	COM_INVALID_COM_PORT	Invalid COM port
-65	COM_USB_DEVICE_NOT_FOUND	USB device not found
-66	COM_NO_USB_DRIVER	No USB driver installed
-67	COM_USB_NOT_SUPPORTED	USB is not supported

#### 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 given 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 given to function is 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 given to 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 given to 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 given to SPP or SPP? is not valid
-1049	PI_STAGE_NAME_ISNT_UNIQUE	A stage name given to 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_COM	WaveEditor: Graph display

	PONENT	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 see the documentation to determine the required device driver version.
-1075	PI_INVALID_LIBRARY_VERSION	The library used doesn't match the required version. Please see 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 <a href="http://www.pi.ws">www.pi.ws</a> .

-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	Switching off the servo mode has failed.
-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.
-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
-1112	PI_ERROR_CANNOT_DETERMINE_ACTUAL_END_OF_TRAVEL_WHILE_PLATFORM_IS_MOVING	Cannot determine actual end of travel range while platform is moving.
-1113	PI_ERROR_AT_QIDN	Sending IDN? or receiving the response has failed.
-1114	PI_ERROR_AT_MAC_DEF	Sending MAC_DEF or receiving the response has failed.
-1115	PI_CONTROLLER_OR_CONTROLLER_VERSION_DOES_NOT_EXIST_IN_PISTAGES_DATABASE	Sending Controller or controller version does not exist in PISTages database.
-1116	PI_NOT_ENOUGH_MEMORY	Not enough memory
-1117	PI_ERROR_AXIS_RUNTIME_ERROR	Runtime error indicated for axis, check error log with \"LOG?\" to find more details.
-1118	PI_ERROR_SYSTEM_RUNTIME_CRITICAL_ERROR	Critical error indicated for system, check error log with \"LOG?\" to find more details.

-1119	PI_ERROR_CANNOT_START_EMULATOR	Cannot start emulation software.
-1120	COM_DEVICE_NOT_SUPPORTED	Device is not supported
-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_CONTAINS_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.
-10016	PI_PARAMETER_DB_MISSING_FIRMWARE_FEATURE_ON_CONTROLLER	Parameters could not be set on controller because the corresponding firmware feature is missing



## 9 Adapting Settings

### 9.1 Settings of the C-877

The properties of the C-877 and the connected positioner are stored in the C-877 as parameter values (e.g., settings for the servo algorithm (p. 20)).

The 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

The write permission for the parameters is determined by command levels.

Every parameter is in the volatile as well as in the nonvolatile memory of the C-877. The values in the nonvolatile memory are loaded to the volatile memory as default values when switching on or rebooting the C-877. The values in the volatile memory determine the current behavior of the system.

The designation "Active Values" is used for the parameter values in the volatile memory and "Startup Values" is used for the parameter values in the nonvolatile memory in the PC software from PI.

### 9.2 Changing Parameter Values in the C-877

#### NOTICE



#### Unsuitable parameter settings!

The values in the nonvolatile memory are loaded to the volatile memory as default values when switching on or rebooting the C-877 and take effect immediately. Unsuitable parameter settings can cause damage to the connected mechanics.

- Change parameter values only after careful consideration.
- Save the current parameter values to the PC (p. 165) before you make changes in the nonvolatile memory.

#### INFORMATION

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

- Overwrite the default values only when it is necessary.
- Save the current parameter values to the PC (p. 165) before you make changes in the nonvolatile memory.
- Contact our customer service department (p. 191), if the C-877 exhibits unexpected behavior.

**INFORMATION**

If the connected positioner has an ID chip (p. 11), the data is loaded from the ID chip into the volatile memory of the C-877 after switching on or rebooting the C-877.

The ID chip only contains some of the information that is required to operate the positioner with the C-877. When you use the PC software from PI, further information is loaded as parameter values from a positioner database (p. 11) into the volatile memory of the C-877.

Parameters that are loaded from the ID chip or from a positioner database are marked in color in the parameter overview (p. 171).

### 9.2.1 General Commands for Parameters

The following general commands are available for parameters:

Command	Function
CCL	Change to a higher command level, e.g., to obtain write permission for particular parameters.
CCL?	Get active command level.
HPA?	Responds with a help string that contains all available parameters with short descriptions.
RPA	Copy a parameter value from the nonvolatile to the volatile memory.
SEP	Change parameters in the nonvolatile memory.
SEP?	Get parameter values from the nonvolatile memory.
SPA	Change parameters in the volatile memory.
SPA?	Get parameter values from the volatile memory.
WPA	Copy a current parameter value from the volatile to the nonvolatile memory. Here it is used as a default value.

You can find details in the command descriptions (p. 82).

### 9.2.2 Commands for Fast Access to Individual Parameters

The following special commands only change the corresponding parameters in the volatile memory. When necessary, the changed values must be written to the nonvolatile memory with the WPA command (p. 138).

**INFORMATION**

The parameters listed below can also be changed with the general commands.

Command	Adaptable parameters
ACC	Acceleration in closed-loop operation (0xB)
DEC	Deceleration in closed-loop operation (0xC)
VEL	Velocity in closed-loop operation (0x49)

You can find details in the command descriptions (p. 82).

### 9.2.3 Saving Parameter Values in a Text File

#### INFORMATION

The C-877 is configured via parameters, e.g., to adapt the mechanics connected. Changing parameter values can cause undesirable results.

- Create a backup copy on the PC before changing the parameter settings of the C-877. You can then restore the original settings at any time.
- Create an additional backup copy with a new file name each time after optimizing the parameter values or adapting the C-877 to specific mechanics.

#### INFORMATION

Parameter values saved in a text file on the PC can be loaded back to the C-877 in PIMikroMove or PITerminal. The **Send file...** button is available for this purpose in the send command window. Before loading into the C-877, the individual lines of the text files must be converted into command lines that contain the corresponding SPA or SEP commands.

#### Requirements

- ✓ You have established communication with PIMikroMove or PITerminal between the C-877 and the PC (p. 50).

#### 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 the **F4** key on the keyboard.

In PITerminal the main window from which commands can be sent is opened automatically after establishing communication.
2. Get the parameter values from which you want to create a backup copy.
  - If you want to save the parameter values from the volatile memory of the C-877: Send the **SPA?** command.
  - If you want to save the parameter values from the nonvolatile memory of the C-877: Send the **SEP?** command.
3. Click on the **Save...** button.

The **Save content of terminal as textfile** window opens.

4. Save the queried parameter values in a text file to your PC in the **Save content of terminal as textfile** window.

### 9.2.4 Changing Parameter Values: General Procedure

For working with parameters, you can use the general commands (p. 164) and the commands for quick access (p. 164).

For simpler access to parameters, PIMikroMove is used in the following, so you do not have to deal with the corresponding commands.

#### NOTICE



##### Unsuitable parameter settings!

The values in the nonvolatile memory are loaded to the volatile memory as default values when switching on or rebooting the C-877 and take effect immediately. Unsuitable parameter settings can cause damage to the connected mechanics.

- Change parameter values only after careful consideration.
- Save the current parameter values to the PC (p. 165) before you make changes in the nonvolatile memory.

#### INFORMATION

The following procedure is generally recommended for changing parameter values:

1. Change the parameter values in the volatile memory.
2. Check whether the C-877 works correctly with the changed parameter values.

If so:

- Write the changed parameter values into the nonvolatile memory.

If not:

- Change and check the parameter values in the volatile memory again.

#### INFORMATION

The write access for the parameters of the C-877 is defined by command levels. After the controller is switched on or rebooted, the active command level is always 0. On command levels > 1, write access is only available to PI service personnel.

- Contact the customer service department if there seem to be problems with parameters of command level 2 or higher (p. 191).

#### Requirements

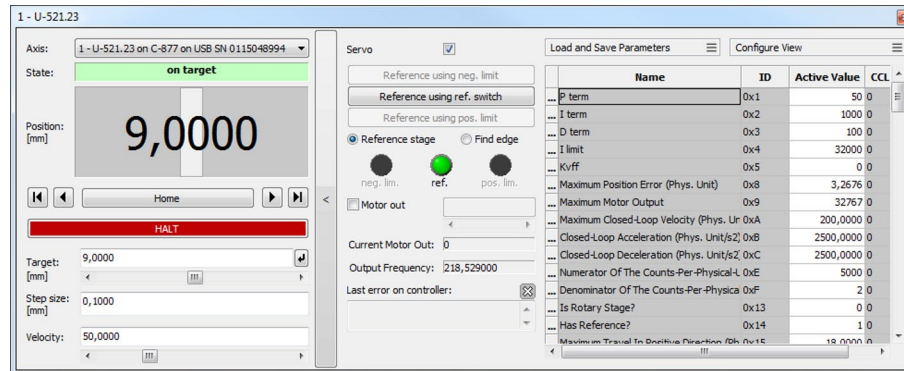
- ✓ If you want to change parameter values in the nonvolatile memory of the C-877: You have saved the parameter values of the C-877 in a text file on the PC (p. 165).
- ✓ You have established communication between the C-877 and the PC with PIMikroMove (p. 50).

### Changing parameter values: General procedure

1. Display the parameter list in PIMikroMove.

If you want to change the axis-related parameters of the C-877:

- a) Open the expanded single axis window for the connected positioner in the main window of PIMikroMove by clicking the right mouse button on the corresponding line of the **Axes** tab and selecting **Show Expanded Single Axis Window** in the context menu.



- b) If the parameter to be modified is not included in the list on the right-hand side of the window, click **Configure View > Select parameters...** and add it to the list. You can also display certain groups of parameters or all axis-related parameters.

If you want to change the system-related parameters of the C-877:

- Open the window for the system-related parameters of the C-877 in the main window of PIMikroMove by selecting **C-877 > Show system parameters** in the menu.

2. Change the desired parameter values in the volatile or nonvolatile memory of the C-877 in the corresponding parameter list.

If you want to change parameter values in the volatile memory, you have the following options:

- Type the new parameter value into the corresponding input field in the **Active Value** column of the list. Press the **Enter** key on the PC keyboard or click with the mouse outside the input field to transfer the parameter value to the volatile memory of the C-877.
- Click **Load and Save Parameters -> Load all startup parameters of the axis / system from controller** in order to load the values of all axis-related / system-related parameters from the nonvolatile memory of the C-877.
- Click **Load and Save Parameters > Load parameters from stage database...** in the extended single-axis window to load a selected parameter set for the axis from the positioner database. You can use **Load and Save Parameters > Reload parameters from stage database...** to reload the currently loaded parameter set.

If you want to change parameter values in the nonvolatile memory, you have the following options:

- Type the new parameter value into the appropriate input field in the **Startup Value** column in the list. Press the **Enter** key on the PC keyboard or click with the mouse outside the input field to transfer the parameter value to the nonvolatile memory of the C-877.
- Click **Load and Save Parameters -> Save all currently active axis / system parameters as startup parameters to controller** to write the values of all axis-related / system-related parameters from the volatile to the nonvolatile memory of the C-877. You can skip parameters that do not have write access on the current command level.

If a parameter value in the volatile memory (**Active Value** column) is different from the parameter value in the nonvolatile memory (**Startup Value** column), the line in the list is highlighted in color.

### 9.3 Creating or Changing a Positioner Type

You can select a suitable parameter set for your positioner from a positioner database in the PC software from PI. The software transfers the values of the selected parameter set to the volatile or nonvolatile memory of the controller. For further information, see "Positioner Databases" (p. 11).

You can create and edit new parameter records in the PI\_UserStages2.dat positioner database. This can be required in the following cases, for example:

- You would like to operate a positioner with different servo control parameter settings than those in the default parameter set.
- You would like to adapt the soft limits of the positioner to your application.
- You have a custom positioner.

#### INFORMATION

You can create a new positioner type easily by modifying an existing positioner type in PIMikroMove and saving it under a new name.

#### INFORMATION

If a positioner type with the same name is present in the standard positioner database (PIStages2.dat or PIMicosStages2.dat) and in the PI\_UserStages2.dat database, the parameter settings from the standard database will always be loaded when this positioner type is selected in the PC software. The parameter settings from PI\_UserStages2.dat are not used in this case.

- When saving positioner types, only assign names that are **not** already used in the PIStages2.dat or PIMicosStages2.dat positioner database.

In the following, PIMikroMove is used for creating a new positioner type and for modifying an existing positioner type.

### Requirements

- ✓ You have installed the latest version of the pistages2.dat and pimicosstages2.dat positioner databases onto your PC (p. 45).
- ✓ If PI has provided you with a custom database for your positioner, then you have installed this database on your PC (p. 46).
- ✓ You have established communication with PIMikroMove between the C-877 and the PC (p. 50).

### Creating a positioner type in the positioner database

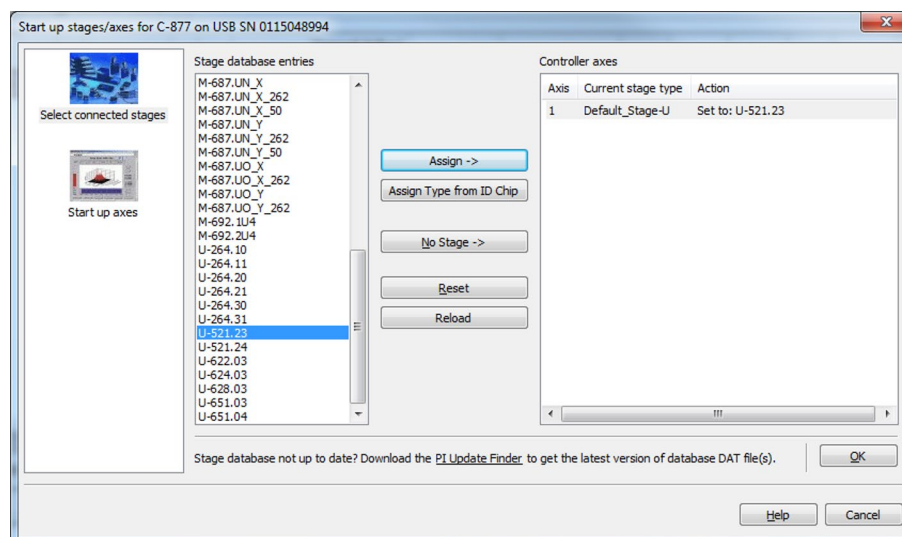
1. In the main window of PIMikroMove, select the **C-877 > Select connected stages...** menu item.

The **Start up stages/axes for C-877** window opens, the step **Select connected stages** is active.

2. Select an appropriate type of positioner during the **Select connected stages** step:
  - Click on **Assign Type from ID Chip**.

or

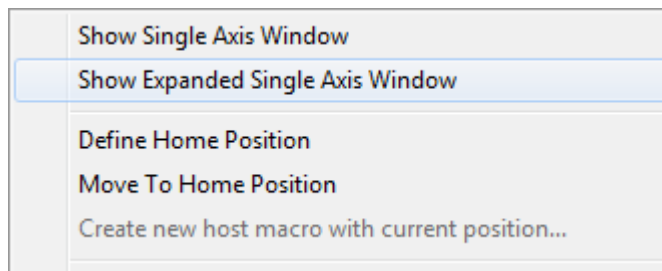
- a) Mark the positioner type in the **Stage database entries** list.
- b) Click **Assign**.



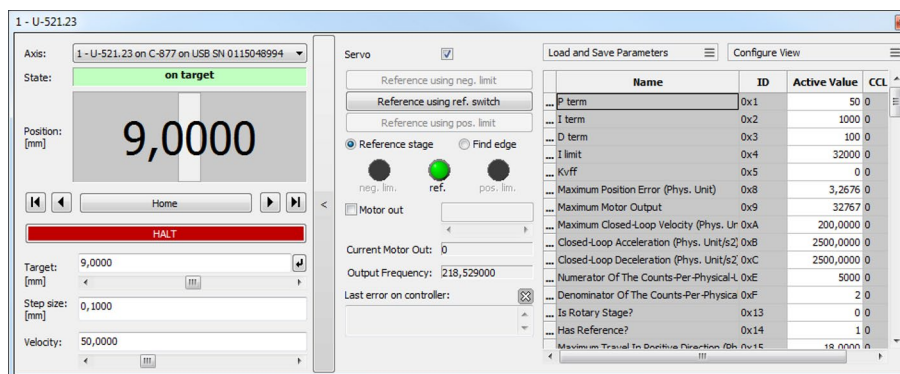
- c) Confirm the selection with **OK**.
3. In the **Save all changes permanently** dialog, click **Keep the changes temporarily** to load the parameter settings into the volatile memory of the C-877.

The **Start up stages/axes** window changes to the step **Start up axes**.

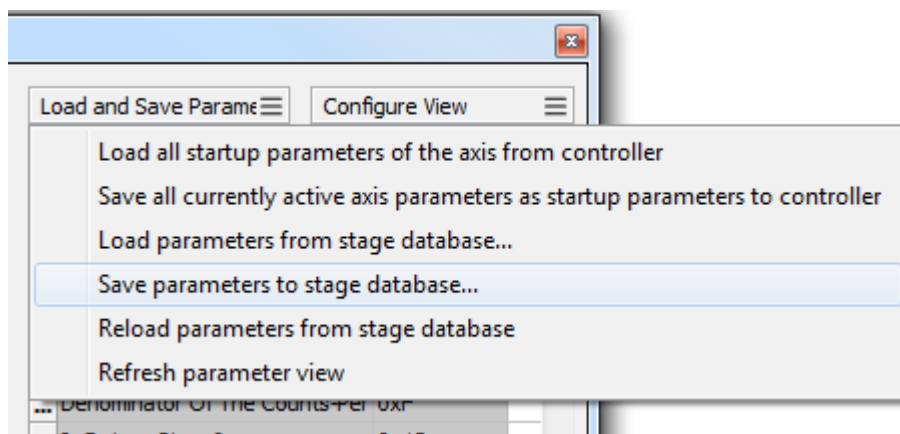
4. In the step **Start up axes** click on **Close** to close the **Start up stages/axes** window.
5. Open the expanded single axis window for the selected positioner in the main window of PIMikroMove by clicking the right mouse button on the corresponding line of the **Axes** tab and selecting **Show Expanded Single Axis Window** in the context menu.



6. Enter new values for the parameters to be changed:



- If the parameter to be modified is not included in the list on the right-hand side of the window, click on **Configure view > Select parameters...** and add it to the list. You can also display certain groups of parameters or all axes-related parameters.
  - Type the new parameter value into the corresponding input field in the **Active Value** column of the list.
  - Press the **Enter** key on the PC keyboard or click outside the input field with the mouse to transfer the parameter value to the volatile memory of the controller. Note: If a parameter value in the volatile memory (**Active Value** column) is different to the parameter value in the nonvolatile memory (**Startup Value** column), the line in the list is highlighted in color.
7. Click on **Load and Save Parameters -> Save parameters to stage database...**



The **Save Parameters as User Stage Type** dialog opens.

8. Save the changed parameter values as new positioner type in the **Save Parameters as User Stage Type** dialog:



- a) Leave the entry in the **Parameters of axis** field unchanged.
- b) Enter the name for the new positioner type into the **Save as** field.
- c) Click **OK**.

The new positioner type was saved to the PI\_UserStages2.dat positioner database. The display of the connected positioner type was updated in the single axis window and in the main window of PIMikroMove. The new positioner type is also available immediately for selection in the **Select connected stages** step.

### Changing a positioner type in the positioner database

1. Select the **C-877 > Select connected stages...** menu item in the main window of PIMikroMove.  
The **Start up stages/axes for C-877** window opens, the **Select connected stages** step is active.
2. Select one of the positioners you created as described above (p. 169): Proceed with the selection as described in step 2 of the **Creating a positioner type in the positioner database** instruction.
3. Proceed with steps 3 to 7 in **Creating a positioner type in the positioner database**.
4. Save the modified parameter values of the positioner type in the **Save Parameters as User Stage Type** dialog:
  - a) Leave the entry in the **Parameters of axis** field unchanged.
  - b) Leave the entry in the **Save as** field unchanged.
  - c) Click **OK**.
  - d) Click **Change settings** in the **Stage type already defined** dialog. The **Save Parameters as User Stage Type** dialog closes automatically after a short time.

The parameter values of the positioner type have been updated in the PI\_UserStages2.dat positioner database and in the main window of PIMikroMove.

## 9.4 Parameter Overview

### INFORMATION

The write access for the parameters of the C-877 is defined by command levels. After the controller is switched on or rebooted, the active command level is always 0. On command levels > 1, write access is only available to PI service personnel.

- Contact the customer service department if there seem to be problems with parameters of command level 2 or higher (p. 191).

### INFORMATION

The password for saving the parameter values in the nonvolatile memory is 100.

Meaning of the color emphasis in the parameter table:

Dark gray:	The value of the parameter is loaded from the ID chip of the positioner (p. 11).
Light gray:	The value of the parameter can be loaded from a positioner database (p. 11).
Colorless:	The value of the parameter can only be changed via command (SPA, SEP, special commands (p. 164)) or by using corresponding operating elements of the PC software (p. 166).

Designations in the header of the following table:

- ID = Parameter ID, hexadecimal format
- Type = Data type:
  - INT = Integer value, including Boolean values
  - FLOAT = Floating point number
  - CHAR = String format
- CL = Command Level for write access
- Element = Element type to which the parameter refers, for more information, see "Commandable Items" (p. 13)
- Parameter name = Name of the parameter
- Description = Explanation of the parameter

ID	Type	CL	Item	Parameter Name	Description
0x1	INT	0	Axis	P term	Present for compatibility reasons only The values of these parameters are set automatically to the values of parameters 0x411, 0x412, 0x413, 0x414, and 0x415.
0x2	INT	0	Axis	I term	
0x3	INT	0	Axis	D term	
0x4	INT	0	Axis	I limit	
0x5	INT	0	Axis	Kvff	
0x8	FLOAT	0	Axis	Maximum Position Error (Phys. Unit)	Maximum position error Is used for the detection of motion errors. For details, see "Behavior with Motion Errors" (p. 61).
0x9	INT	0	Axis	Maximum Motor Output	Maximum permissible absolute measure of the control value (dimensionless) For details see "Supported Motor Types" (p. 28).

ID	Type	CL	Item	Parameter Name	Description
0xA	FLOAT	0	Axis	Maximum Closed Loop Velocity (Phys. Unit/s)	Maximum velocity in closed-loop operation Specifies the maximum value for parameter 0x49.
0xB	FLOAT	0	Axis	Closed Loop Acceleration (Phys. Unit/s <sup>2</sup> )	Acceleration in closed-loop operation For details, see "Generation of Dynamics Profile" (p. 18). Is limited by parameter 0x4A.
0xC	FLOAT	0	Axis	Closed-Loop Deceleration (Phys. Unit/s <sup>2</sup> )	Deceleration in closed-loop operation For details, see "Generation of Dynamics Profile" (p. 18). Is limited by parameter 0x4B.
0xE	INT	0	Axis	Numerator Of The Counts-Per-Physical-Unit-Factor	Numerator and denominator of the factor for counts per physical length unit For details, see "Physical Units" (p. 16).
0xF	INT	0	Axis	Denominator Of The Counts-Per-Physical-Unit-Factor	
0x13	INT	0	Axis	Is Rotary Stage?	Is this a rotation stage? 0 = No rotation stage 1 = Rotation stage No evaluation by the C-877, but only by the PC software: PIMikroMove determines on the basis of this value which motion is permitted.
0x14	INT	0	Axis	Has Reference?	Does the positioner have a reference switch? For details, see "Reference Switch Detection" (p. 29).
0x15	FLOAT	0	Axis	Maximum Travel In Positive Direction (Phys. Unit)	Soft limit in positive direction See examples in "Travel Range and Soft Limits" (p. 32).
0x16	FLOAT	0	Axis	Value At Reference Position (Phys. Unit)	Position value on the reference switch See examples in "Travel Range and Soft Limits" (p. 32).
0x17	FLOAT	0	Axis	Distance From Negative Limit To Reference Position (Phys. Unit)	Distance between reference switch and negative limit switch See examples in "Travel Range and Soft Limits" (p. 32).

ID	Type	CL	Item	Parameter Name	Description
0x18	INT	0	Axis	Limit Mode	Signal logic of the limit switches For details, see "Limit Switch Detection" (p. 30).
0x2F	FLOAT	0	Axis	Distance From Reference Position To Positive Limit (Phys. Unit)	Distance between reference switch and positive limit switch See examples in "Travel Range and Soft Limits" (p. 32).
0x30	FLOAT	0	Axis	Maximum Travel In Negative Direction (Phys. Unit)	Soft limit in negative direction See examples in "Travel Range and Soft Limits" (p. 32).
0x31	INT	0	Axis	Invert Reference?	Should the reference signal be inverted? For details, see "Reference Switch Detection" (p. 29).
0x32	INT	0	Axis	Has No Limit Switches?	Does the positioner not have limit switches? For details, see "Limit Switch Detection" (p. 30).
0x33	INT	0	Axis	Motor Offset Positive	Offset for the positive direction of motion For details, see "Servo Algorithm and Other Control Value Corrections" (p. 20).
0x34	INT	0	Axis	Motor Offset Negative	Offset for the negative direction of motion For details, see "Servo Algorithm and Other Control Value Corrections" (p. 20).
0x36	INT	0	Axis	Settling Window (encoder counts)	Present for compatibility reasons only The value of this parameter is set automatically to the value of parameter 0x406.
0x3C	CHAR	0	Axis	Stage name	Positioner name Maximum of 20 characters; default value: NOSTAGE The value NOSTAGE "deactivates" the axis. A deactivated axis is not accessible for axis-related commands (e.g., motion commands or position queries).

ID	Type	CL	Item	Parameter Name	Description
0x3F	FLOAT	0	Axis	Settling Time (s)	Delay time for setting the on-target state. For details, see "On-Target State" (p. 27).
0x47	INT	0	Axis	Reference Travel Direction	Default direction for the reference move For details, see "Referencing" (p. 34).
0x48	INT	0	Axis	Motor Drive Offset	Velocity-dependent offset For details, see "Servo Algorithm and Other Control Value Corrections" (p. 20).
0x49	FLOAT	0	Axis	Closed-Loop Velocity (Phys. Unit/s)	Velocity in closed-loop operation For details, see "Generating a Dynamics Profile" (p. 18). Is limited by parameter 0xA
0x4A	FLOAT	0	Axis	Maximum Closed-Loop Acceleration (Phys. Unit/s <sup>2</sup> )	Maximum acceleration in closed-loop operation Specifies the maximum value for parameter 0xB.
0x4B	FLOAT	0	Axis	Maximum Closed-Loop Deceleration (Phys. Unit/s <sup>2</sup> )	Maximum deceleration in closed-loop operation Specifies the maximum value for parameter 0xC.
0x4D	INT	0	Axis	Servo Window Mode	Reference variable for position windows for switching between parameter groups For details, see "Servo Algorithm and Other Control Value Corrections" (p. 20).
0x50	FLOAT	0	Axis	Velocity For Reference Moves (Phys. Unit/s)	Maximum velocity for reference move For details, see "Referencing" (p. 34).
0x51	FLOAT	0	Axis	Output Frequency (kHz)	Frequency of the piezo voltage For details see "Supported Motor Types" (p. 28).
0x52	INT	0	Axis	Frequency Control	State of the frequency control For details, see "Frequency Control" (p. 28).
0x53	FLOAT	0	Axis	Minimum Output Frequency (kHz)	Minimum frequency of the piezo voltage For details, see "Frequency Control" (p. 28).

ID	Type	CL	Item	Parameter Name	Description
0x54	FLOAT	0	Axis	Maximum Output Frequency (kHz)	Maximum frequency of the piezo voltage For details, see "Frequency Control" (p. 28).
0x55	INT	0	Axis	Minimum Motor Output For Frequency Control	Minimum control value for activating the frequency control For details, see "Frequency Control" (p. 28).
0x5A	INT	0	Axis	Numerator Of The Servo-Loop Input Factor	Numerator and denominator of the servo-loop input factor For details, see "Servo Algorithm and Other Control Value Corrections" (p. 20).
0x5B	INT	0	Axis	Denominator Of The Servo-Loop Input Factor	
0x5C	INT	0	Axis	Source Of Reference Signal	Reference signal source for the <b>FRF</b> (p. 102) or <b>FED</b> (p. 99) commands
0x5D	INT	0	Axis	Source Of Negative Limit Signal	Reference signal source for the <b>FNL</b> (p. 100) or <b>FED</b> (p. 99) commands
0x5E	INT	0	Axis	Source of Positive Limit Signal	Reference signal source for the <b>FPL</b> (p. 102) or <b>FED</b> (p. 99) commands
0x5F	INT	0	Axis	Invert Digital Input Used For Negative Limit	Inverts the polarity of the digital inputs that are used for the source of the negative limit switch signal.
0x60	INT	0	Axis	Invert Digital Input Used For Positive Limit	Inverts the polarity of the digital inputs that are used for the source of the positive limit switch signal.
0x63	FLOAT	0	Axis	Distance Between Limit And Hard Stop (Phys. Unit)	Distance between internal limit switch and hard stop For details, see "Referencing" (p. 34).
0x64	INT	0	Axis	Phase Shift	Phase shift between current and voltage on the drive
0x70	INT	0	Axis	Reference signal mode	Reference signal type For details, see "Reference Switch Detection" (p. 29).
0x71	INT	0	Axis	D Term Delay (No. Of Servo Cycles)	D term delay For details, see "Servo Algorithm and Other Control Value Corrections" (p. 20).
0x72	INT	0	System	Ignore Macro Error?	Ignore macro error? For details, see "Commands and Parameters for Macros" (p. 65).

ID	Type	CL	Item	Parameter Name	Description
0x77	INT	0	Axis	Use Limit Switches Only For Reference Moves	Should the limit switches only be used for reference moves? For details, see "Limit Switch Detection" (p. 30).
0x78	FLOAT	0	Axis	Distance From Limit To Start Of Ref Search (Phys. Unit)	Distance between the limit switch and the starting position for the reference move to the index pulse For details, see "Referencing" (p. 34).
0x79	FLOAT	0	Axis	Distance For Reference Search (Phys. Unit)	Maximum distance for the reference move to the index pulse For details, see "Referencing" (p. 34).
0x7B	FLOAT	0	Axis	PID Maximum Output Time (s)	Maximum time period for which a high control value can be set in closed-loop operation. For details, see "Protection Against Overheating" (p. 61).
0x7C	FLOAT	0	Axis	Maximum Motor Output (V)	Maximum permissible piezo voltage For details see "Supported Motor Types" (p. 28).
0x400	INT	0	Axis	Number of Servo Parameter Groups	Maximum number of parameter groups used For details, see "Servo Algorithm and Other Control Value Corrections" (p. 20).
0x401	INT	0	Axis	P term 0	Proportional constant of parameter group 0
0x402	INT	0	Axis	I term 0	Integral constant of parameter group 0
0x403	INT	0	Axis	D term 0	Differential constant of parameter group 0
0x404	INT	0	Axis	I limit 0	Limitation of the integral constant of parameter group 0
0x405	INT	0	Axis	Kvff 0	Feed-forward control of the commanded velocity for parameter group 0
0x406	INT	0	Axis	Window Enter 0	Position window for activating parameter group 0 For further details, see "On-Target-Status" (p. 27)

ID	Type	CL	Item	Parameter Name	Description
0x407	INT	0	Axis	Window Exit 0	Position window for deactivating parameter group 0 For further details, see "On-Target-Status" (p. 27)
0x409	INT	0	Axis	2nd Phase On 0 (No. Of Servo Cycles)	Duty cycle of parallel control of both piezo segments for parameter group 0 For details, see "Optional Two-Phase Control" (p. 26)
0x40A	INT	0	Axis	2nd Phase Off 0 (No. Of Servo Cycles)	Pause duration for parallel control of both piezo segments for parameter group 0 For details, see "Optional Two-Phase Control" (p. 26)
0x411	INT	0	Axis	P term 1	Proportional constant of parameter group 1
0x412	INT	0	Axis	I term 1	Integral constant of parameter group 1
0x413	INT	0	Axis	D term 1	Differential constant of parameter group 1
0x414	INT	0	Axis	I limit 1	Limitation of the integral constant of parameter group 1
0x415	INT	0	Axis	Kvff 1	Feed-forward control of the commanded velocity for parameter group 1
0x416	INT	0	Axis	Window Enter 1	Position window for activating the parameter group 1 For further details, see "On-Target-Status" (p. 27)
0x417	INT	0	Axis	Window Exit 1	Position window for deactivating the parameter group 1 For further details, see "On-Target-Status" (p. 27)
0x419	INT	0	Axis	2nd Phase On 1 (No. Of Servo Cycles)	Duty cycle of parallel control of both piezo segments for parameter group 1 For details, see "Optional Two-Phase Control" (p. 26)
0x41A	INT	0	Axis	2nd Phase Off 1 (No. Of Servo Cycles)	Pause duration for parallel control of both piezo segments for parameter group 1 For details, see "Optional Two-Phase Control" (p. 26)
0x421	INT	0	Axis	P term 2	Proportional constant of parameter group 2



ID	Type	CL	Item	Parameter Name	Description
0x422	INT	0	Axis	I term 2	Integral constant of parameter group 2
0x423	INT	0	Axis	D term 2	Differential constant of parameter group 2
0x424	INT	0	Axis	I limit 2	Limitation of the integral constant of parameter group 2
0x425	INT	0	Axis	Kvff 2	Feed-forward control of the commanded velocity for parameter group 2
0x426	INT	0	Axis	Window Enter 2	Position window for activating parameter group 2 For further details, see "On-Target-Status" (p. 27)
0x427	INT	0	Axis	Window Exit 2	Position window for deactivating parameter group 2 For further details, see "On-Target-Status" (p. 27)
0x429	INT	0	Axis	2nd Phase On 2 (No. Of Servo Cycles)	Duty cycle of parallel control of both piezo segments for parameter group 2 For details, see "Optional Two-Phase Control" (p. 26)
0x42A	INT	0	Axis	2nd Phase Off 2 (No. Of Servo Cycles)	Pause duration for parallel control of both piezo segments for parameter group 2 For details, see "Optional Two-Phase Control" (p. 26)
0x431	INT	0	Axis	P term 3	Proportional constant of parameter group 3
0x432	INT	0	Axis	I term 3	Integral constant of parameter group 3
0x433	INT	0	Axis	D term 3	Differential constant of parameter group 3
0x434	INT	0	Axis	I limit 3	Limitation of the integral constant of parameter group 3
0x435	INT	0	Axis	Kvff 3	Feed-forward control of the commanded velocity for parameter group 3
0x436	INT	0	Axis	Window Enter 3	Position window for activating parameter group 3
0x437	INT	0	Axis	Window Exit 3	Position window for deactivating parameter group 3

ID	Type	CL	Item	Parameter Name	Description
0x439	INT	0	Axis	2nd Phase On 3 (No. Of Servo Cycles)	Duty cycle of parallel control of both piezo segments for parameter group 3
0x43A	INT	0	Axis	2nd Phase Off 3 (No. Of Servo Cycles)	Pause duration for parallel control of both piezo segments for parameter group 3
0x441	INT	0	Axis	P term 4	Proportional constant of parameter group 4
0x442	INT	0	Axis	I term 4	Integral constant of parameter group 4
0x443	INT	0	Axis	D term 4	Differential constant of parameter group 4
0x444	INT	0	Axis	I limit 4	Limitation of the integral constant of parameter group 4
0x445	INT	0	Axis	Kvff 4	Feed-forward control of the commanded velocity for parameter group 4
0x446	INT	0	Axis	Window Enter 4	Position window for activating parameter group 4
0x447	INT	0	Axis	Window Exit 4	Position window for deactivating parameter group 4
0x449	INT	0	Axis	2nd Phase On 4 (No. Of Servo Cycles)	Duty cycle of parallel control of both piezo segments for parameter group 4
0x44A	INT	0	Axis	2nd Phase Off 4 (No. Of Servo Cycles)	Pause duration for parallel control of both piezo segments for parameter group 4
0x7000000	FLOAT	0	Axis	Range Limit Min	Additional soft limit for the negative direction of motion (physical unit) For details, see "Travel Range and Soft Limits" (p. 30).
0x7000001	FLOAT	0	Axis	Range Limit Max	Additional soft limit for the positive direction of motion (physical unit) For details, see "Travel Range and Soft Limits" (p. 30).
0x7000601	CHAR	0	Axis	Axis Unit	Unit symbol For details, see "Physical Units" (p. 16).
0xE000200	FLOAT	0	System	Servo Update Time	Servo cycle time in seconds
0xF000100	CHAR	2	Axis	Stage Type	Type of the positioner Form for standard positioners: x-xxx Form for custom positioners: x-xxxKxxx

ID	Type	CL	Item	Parameter Name	Description
0xF000200	CHAR	2	Axis	Stage Serial Number	Serial number of the positioner 9-digit number
0xF000300	CHAR	2	Axis	Stage Assembly Date	Assembly date of the positioner Date format: DDMMYY
0xF000400	INT	2	Axis	Stage HW Version	Version number of the positioner hardware



## 10 Maintenance

### 10.1 Cleaning the C-877

#### NOTICE



##### Short circuits or flashovers!

The C-877 contains electrostatic-sensitive devices that can be damaged by short-circuiting or flashovers when cleaning fluids penetrate the housing.

- Before cleaning, disconnect the C-877 from the power source by removing the mains plug.
- Prevent cleaning fluid from penetrating the housing.

- When necessary, clean the surfaces of the C-877's housing using a cloth dampened with a mild cleanser or disinfectant.

### 10.2 Updating Firmware

#### INFORMATION

The `*IDN?` command reads the version number of the firmware among other things.

Example of a C-877 response:

```
(c)2024 Physik Instrumente (PI) GmbH & Co. KG, C-877.1U11,  
117048994, 01.400
```

- C-877.1U11: Device name
- 117048994: Serial number of the device.
- 01.400: Firmware version

#### INFORMATION

If the C-877 is in firmware update mode, the **STA** LED is off. The C-877 does not leave the firmware update mode until it is **rebooted** after a **successful** firmware update. If the firmware update was unsuccessful or aborted, the C-877 remains in the firmware update mode after a reboot.

If the **STA** LED is still off, even though the C-877 has been rebooted after the firmware update:

- Repeat the firmware update.
- If the update of the firmware fails, contact our customer service department (p. 191).

**INFORMATION**

If new parameters are introduced with the firmware update or the C-877 memory management is changed, an initialization of the C-877 is required after updating the firmware.

**Requirements**

- ✓ You have connected the C-877 to the PC via the USB interface (p. 47).
- ✓ The **PIFirmwareManager** program is installed on the PC (p. 44).
- ✓ You have copied the new firmware file, which you have received from our customer service department, to a directory on the PC.
- ✓ You have read and understood the documentation which you received from our customer service department together with the new firmware. You have learned from the documentation whether new parameters are introduced with the firmware update or the memory management of the C-877 changes.
- ✓ You have saved the parameter values of the C-877 to a text file on the PC (p. 165)
- ✓ You have saved the C-877 controller macros to files on the PC (p. 72).
- ✓ You have established communication with PIMikroMove or PITerminal between the C-877 and the PC (p. 50).

**Updating the firmware of the C-877**

1. Start the **PIFirmwareManager** program on the PC and update the controller firmware. Proceed as described in the user manual SM164E (p. 3).

**Restarting the C-877**

1. Switch off the C-877.
2. Switch the C-877 on again.

If the firmware update was successful, the C-877 exits the firmware update mode and the **STA** LED lights up green.

Have new parameters been added by the firmware update, or has the memory management of the C-877 been changed?

- If no: Firmware update is finished.
- If yes: An initialization of the C-877 is required, see below.

**Initializing the C-877 after a firmware update**

The initialization of the C-877 resets **all** parameters to their factory settings and deletes all controller macros. Consequently, parameter values and controller macros that are not saved are lost during the initialization process.

1. Make sure that the current parameter values and controller macros of the C-877 have been saved on the PC.
2. On the PC, start PITerminal or PIMikroMove, connect to the C-877, and, if necessary, open the window to send commands.
3. Initialize the C-877, by sending the following commands one by one:

```
ZZZ 100 parameter
```

```
ZZZ 100 macros
```

After successful initialization, the controller issues a corresponding message.

4. Adapt the parameter values of the C-877.

For instructions on the general procedure, see "Changing Parameter Values: General Procedure" (p. 166).

- Reset the parameters that were already present prior to the firmware update to the saved values from the text file.
- Set the parameters that were introduced with the firmware update to the appropriate values.

5. If you have saved controller macros on the PC: Load the controller macros back to the C-877, see "Making Backups and Loading Controller Macros" (p. 72).





# 11 Troubleshooting

Fault: Positioner does not move	
Possible causes	Remedial measures
Cable not connected correctly	➤ Check the cable connections.
The positioner has been connected to the switched-on C-877	<p>The sensor electronics in the positioner has not been initialized, and the ID chip of the positioner (p. 11) has not been read out.</p> <p>➤ Switch the C-877 off and on again, or reboot the C-877 with the <b>RBT</b> command or with the corresponding functions of the PC software.</p>
Unsuitable cable used	<p>If unsuitable cables are used, interference can occur in the signal transmission between the positioner and the C-877.</p> <p>➤ If the positioner, cable, and C-877 are marked as a coherent system, replace the system components with other components only after consulting PI.</p> <p>➤ If you need extension cables, contact our customer service department (p. 191).</p>
Positioner or cable is defective	➤ If available, replace the defective positioner with another positioner and test the new combination.
Incorrect configuration	➤ Check the parameter settings of the C-877 with the <b>SPA?</b> (volatile memory) and <b>SEP?</b> (nonvolatile memory) commands. Details on parameter settings see "Adapting Settings" (p. 163).
Incorrect command or incorrect syntax	➤ Send the <b>ERR?</b> command and check the error code that is returned.
Incorrect axis commanded	➤ Make sure that the correct axis identifier is used and that the commanded axis belongs to the correct positioner.

Fault: Positioner performs unintentional motion	
Possible causes	Remedial measures
Startup macro is run	➤ Check whether a macro is specified as startup macro and cancel selection of the startup macro if necessary (p. 66).

Fault: Positioner is oscillating or positions inaccurately	
Possible causes	Remedial measures
The load was changed.	➤ Readjust the system according to the changed load (p. 56).

Fault: Positioner is already oscillating during the reference move	
Possible causes	Remedial measures
Very high load on the positioner	<p>In case of a very high load, proceed with PIMikroMove during the reference move as follows:</p> <ol style="list-style-type: none"> <li>1. Do <b>not</b> start the reference move in the <b>Start up axes</b> step, but click on <b>Close</b> to close the <b>Start up controller</b> window instead.</li> <li>2. In the main window, open the single axis window for the positioner connected by selecting the positioner in the <b>View &gt; Single Axis Window</b> menu.</li> <li>3. Expand the view of the single axis window by clicking on the &gt; button at the right edge of the window.</li> <li>4. With the <b>Servo</b> check box, make sure that the servo mode is switched on.</li> <li>5. Start the reference move by clicking on one of the <b>Reference...</b> buttons.</li> <li>6. If the positioner is oscillating: Stop the reference move immediately in the <b>Reference Axes</b> dialog, close the dialog and switch off the servo mode by removing the tick from the respective check box in the single axis window.</li> <li>7. Enter new values for the servo control parameters, see "Optimizing the Servo Control Parameters" (p. 56).</li> <li>8. Restart the reference move.</li> <li>9. If the positioner is still oscillating, repeat steps 6 to 8 until the reference move has completed successfully without oscillation.</li> </ol>

Fault: There is no communication between the controller and the PC	
Possible causes	Remedial measures
The wrong communication cable is used or it is defective	➤ Check whether the USB cable works on a fault-free system.
Another program is accessing the interface.	➤ Close the other program.

Fault: There is no communication between the controller and the PC	
Possible causes	Remedial measures
Problems with special software	<ul style="list-style-type: none"> <li>➤ Check whether the system works with other software, such as a terminal program or a development environment. You can test the communication by starting a terminal program (such as for example, PITerminal) and entering <code>*IDN?</code> or <code>HLP?</code>.</li> <li>➤ Make sure that you end the commands with an LF (line feed). A command is only executed when the LF has been received.</li> </ul>

Fault: The customer software does not function with the PI drivers	
Possible causes	Remedial measures
Incorrect combination of driver routines/VIs	<ul style="list-style-type: none"> <li>➤ Check whether the system functions with a terminal program (e.g., PITerminal). If so:</li> <li>➤ Read the information in the corresponding software manual and compare your program code with the sample code on the data storage device with the PI Software Suite.</li> </ul>

If the problem that occurred with your system is not in the list above or cannot be solved as described, contact our customer service department (p. 191).



## 12 Customer Service Department

For inquiries and orders, contact your PI representative 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. 3).



## 13 Technical Data

Subject to change. You can find the latest product specifications on the product web page at [www.pi.ws](http://www.pi.ws) (<https://www.physikinstrumente.com/en/>).

### 13.1 Specifications




#### 13.1.1 Data Table




<b>Motion and control</b>	<b>C-877.1U11</b>
Axes	1
Servo cycle time	100 $\mu$ s
Profile generator	Trapezoidal velocity profile, point-to-point motion
Encoder input	A/B quadrature TTL level, differential according to RS-422
Stall detection	Servo off, triggered by programmable position error
Limit switches	2 $\times$ TTL (programmable polarity)
Reference switch	1 $\times$ TTL
<b>Electrical properties</b>	<b>C-877.1U11</b>
Max. output power per channel	15 W
Max. output voltage per channel	200 V <sub>pp</sub> , 71 V <sub>eff</sub>
<b>Interfaces and operation</b>	<b>C-877.1U11</b>
Communication interfaces	USB
Motor connector	Sub-D 15 (f)
Command set	PI General Command Set (GCS)
User software	PIMikroMove, PITerminal
Software drivers	NI LabVIEW driver, dynamic libraries for Windows and Linux
Supported functions	Startup macro; data recorder for recording operating data such as motor voltage, velocity, position or position error; internal safety circuitry: Watchdog timer; ID chip detection
Manual control	-

<b>Miscellaneous</b>	<b>C-877.1U11</b>
Operating voltage	24 V DC from external power supply (included in scope of delivery)
Max. current consumption	300 mA plus motor current (max. 0.8 A)
Operating temperature range	5 to 40 °C
Mass	0.13 kg
Dimensions	95 mm × 71 mm × 24 mm (incl. mounting rails)

### 13.1.2 Maximum Ratings

The C-877 is designed for the following operating data:

Input on:	Maximum operating voltage	Operating frequency	Maximum current consumption
			
Barrel connector socket (m)	30 V	==	0.8 A

Output on:	Maximum piezo voltage	Maximum frequency of the piezo voltage	Maximum output current
			
Sub-D 15 (f)	200 V <sub>pp</sub>	500 kHz	280 mA <sub>pp</sub>

### 13.1.3 Ambient Conditions and Classifications

The following ambient conditions and classifications for the C-877 must be observed:

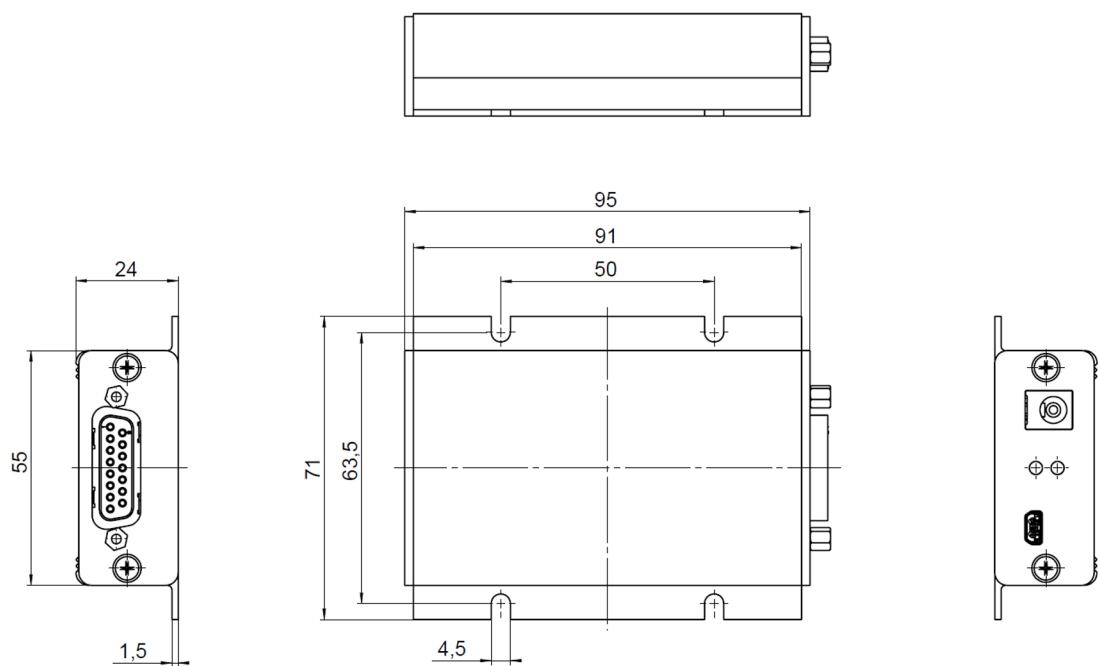
Area of application	For indoor use only
Maximum altitude	2000 m
Air pressure	1100 hPa to 0.1 hPa (corresponds roughly to 825 torr to 0.075 torr)
Relative humidity	Highest relative humidity 80 % for temperatures up to 31 °C Decreasing linearly to 50 % relative humidity at 40 °C
Storage temperature	0 °C to 70 °C
Transport temperature	−25 °C to +85 °C
Overvoltage category	II
Protection class	I



Degree of pollution	2
Measurement category	I
Degree of protection according to IEC 60529	IP20

## 13.2 Dimensions

Dimensions in mm. Note that the decimal points are separated by a comma in the drawings.



### 13.3 Pin Assignment

#### 13.3.1 Motor and Sensor

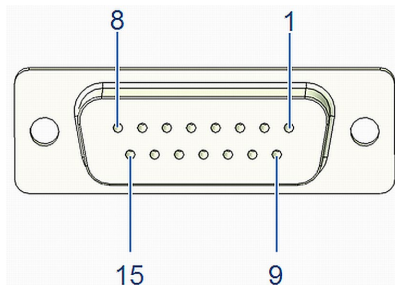



Figure 15: D-sub 15 socket

Pin	Signal	Function
1	NC	Not connected
9	MOTOR GND	Output: piezo
2	MOTOR GND	Output: piezo
10	GND	0 V
3	MOTOR OUT 1	Output: piezo
11	MOTOR OUT 1	Output: piezo
4	VDD	Output: +5 V
12	NLIMIT	Input: negative limit switch, TTL
5	PLIMIT	Input: positive limit switch, TTL
13	REFSWITCH	Input: reference switch, TTL
6	ID CHIP	Bidirectional: ID chip
14	ENCA+	Input: encoder channel A, RS-422
7	ENCA-	Input: encoder channel A (inverted), RS-422
15	ENCB+	Input: encoder channel B, RS-422
8	ENCB-	Input: encoder channel B (inverted), RS-422

#### 13.3.2 Power supply connection

Barrel connector socket (m)

	Pin	Signal	Direction
	Center pin	+24 V DC supply voltage	Input
	Outer conductor	GND	GND

## 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 device according to international, national, and local rules and regulations.

To fulfill the responsibility as the product manufacturer, Physik Instrumente (PI) SE & Co. KG undertakes environmentally correct disposal of all old PI equipment made available on the market after 13 August 2005 without charge.

If you have an old device from PI, you can send it to the following address free of charge:

Physik Instrumente (PI) SE & Co. KG  
Auf der Römerstraße 1  
76228 Karlsruhe, Germany

