

**MS241E**  
**C-663 Mercury Step stepper motor  
controller**

**User Manual**

Version: 1.4.0

Date: 09.07.2020



This document describes the following product:

- **C-663.12**  
Compact Mercury Step stepper motor controller, 1 axis, closed-loop and open-loop operation, HD D-sub 26, 48 V



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

PI®, NanoCube®, PICMA®, PILINE®, NEXLINE®, PiezoWalk®, NEXACT®, Picoactuator®, PI nano®, PI Mag®, Q-Motion®

Notes on brand names and third-party trademarks:

Microsoft® and Windows® are registered trademarks or trademarks of Microsoft Corporation in the USA and/or other countries.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

TwinCAT® is a registered trademark of and licensed by Beckhoff Automation GmbH.

LabVIEW, National Instruments and NI are trademarks of National Instruments. Neither the driver software nor the software programs offered by PI or other goods and services are connected to or sponsored by National Instruments.

Python® is a registered trademark of Python Software Foundation.

BiSS is a registered trademark of iC-Haus GmbH.

The following designations are protected company names, trademarks or registered trademarks of other owners:

Linux, MATLAB, MathWorks

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

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

© 2020 Physik Instrumente (PI) GmbH & Co. KG, Karlsruhe, Germany. The text, photographs, and drawings in this manual are protected by copyright. Physik Instrumente (PI) GmbH & 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: 09.07.2020

Document number: MS241E, ASt, Version 1.4.0

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 This manual is superseded by any new release.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

# 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 .....	4
1.6	Downloading Manuals.....	4
<b>2</b>	<b>Safety</b>	<b>7</b>
2.1	Intended Use .....	7
2.2	General Safety Instructions .....	7
2.3	Organizational Measures.....	8
<b>3</b>	<b>Product Description</b>	<b>9</b>
3.1	Product View .....	9
3.1.1	Front Panel .....	9
3.1.2	Rear Panel.....	10
3.1.3	Type Plate .....	11
3.2	Scope of Delivery.....	12
3.3	Accessories .....	12
3.4	Overview of PC Software.....	13
3.5	Positioner Databases.....	15
3.6	Communication Interfaces .....	16
3.7	Functional Principles .....	17
3.7.1	Block Diagram .....	17
3.7.2	Commandable Elements.....	18
3.7.3	Important Components of the Firmware .....	19
3.7.4	Operating Modes.....	20
3.7.5	Physical Units.....	22
3.7.6	Motion Triggering .....	24
3.7.7	Generating a Dynamics Profile .....	25
3.7.8	Position Control .....	28
3.7.9	Motor Control.....	31
3.7.10	On-Target State .....	32
3.7.11	Reference Switch Detection .....	33
3.7.12	Limit Switch Detection.....	33
3.7.13	Travel Range and Soft Limits .....	34
3.7.14	Referencing.....	38

<b>4</b>	<b>Unpacking</b>	<b>43</b>
<hr/>		
<b>5</b>	<b>Installation</b>	<b>45</b>
5.1	Installing the PC Software .....	45
5.1.1	Doing Initial Installation.....	45
5.1.2	Installing Updates .....	46
5.1.3	Installing Custom Positioner Databases .....	48
5.2	Mounting the C-663 .....	49
5.3	Grounding the C-663 .....	50
5.4	Connecting the Positioner .....	50
5.5	Connecting the PC .....	51
5.5.1	Connecting to the RS-232 Interface .....	52
5.5.2	Connecting to the USB Interface .....	52
5.5.3	Building a Daisy Chain Network.....	53
5.6	Connecting the Power Supply to the C-663 .....	54
5.7	Connecting an Analog Joystick .....	54
5.8	Connecting Digital Inputs and Outputs .....	55
5.8.1	Connecting the Digital Outputs .....	55
5.8.2	Connecting the Digital Inputs .....	56
5.9	Connecting Analog Signal Sources.....	57
<hr/>		
<b>6</b>	<b>Startup</b>	<b>59</b>
6.1	General Notes on Startup.....	59
6.2	Adapting the DIP Switch Settings .....	60
6.2.1	General Procedure.....	60
6.2.2	Controller Address.....	61
6.2.3	Baud Rate.....	62
6.3	Switching the C-663 On .....	62
6.4	Establishing Communication .....	63
6.4.1	Establishing Communication via RS-232.....	63
6.4.2	Establishing Communication via USB .....	64
6.4.3	Establishing Communication for Networked Controllers.....	66
6.5	Starting Motion .....	71
6.6	Optimizing the Servo Control Parameters .....	76
6.7	Configuration Steps for Advanced Users.....	82
6.7.1	Setting the Operating Mode.....	82
6.7.2	Open-loop Operation with Sensor: Position Adjustment.....	83
<hr/>		
<b>7</b>	<b>Operation</b>	<b>85</b>
7.1	Protective Functions of the C-663.....	85
7.1.1	Behavior During System Errors.....	85
7.1.2	Behavior with Motion Errors .....	85
7.1.3	Restoring Operational Readiness .....	86

7.2	Data Recorder.....	87
7.2.1	Configuring the Data Recorder.....	87
7.2.2	Starting the Recording.....	88
7.2.3	Reading Recorded Data .....	88
7.3	Digital Output Signals .....	89
7.3.1	Commands for Digital Outputs .....	89
7.3.2	Configuring the "Position Distance" Trigger Mode .....	90
7.3.3	Configuring the "On Target" Trigger Mode .....	92
7.3.4	Configuring the "Motion Error" Trigger Mode .....	93
7.3.5	Configuring the "In Motion" Trigger Mode .....	93
7.3.6	Configuring the "Position + Offset" Trigger Mode .....	94
7.3.7	Configuring the "Single Position" Trigger Mode.....	95
7.3.8	Setting Signal Polarity .....	96
7.4	Digital Input Signals .....	96
7.4.1	Commands and Parameters for Digital Inputs .....	97
7.4.2	Using Digital Input Signals in Macros .....	99
7.4.3	Using Digital Input Signals as Switch Signals .....	99
7.5	Analog Input Signals .....	101
7.5.1	Commands for Analog Inputs .....	101
7.5.2	Using Analog Input Signals in Macros.....	102
7.6	Joystick Control .....	102
7.6.1	How Joystick Control Works.....	102
7.6.2	Commands and Parameters for Joystick Control .....	103
7.6.3	Controlling Axis Motion .....	104
7.6.4	Calibrating the Joystick .....	105
7.6.5	Joysticks Available .....	108
7.7	Controller Macros.....	110
7.7.1	Overview: Macro Functionality and Example Macros.....	110
7.7.2	Commands and Parameters for Macros.....	111
7.7.3	Working with Macros .....	112
7.7.4	Making Backups and Loading Controller Macros .....	120
7.7.5	Macro Example: Synchronization of Two Controllers .....	121
7.7.6	Macro Example: Stopping Motion by Pushbutton .....	122
7.7.7	Macro Example: Joystick Control with Storage of Positions .....	123
7.7.8	Macro Example: Joystick Control with Change in Velocity.....	127
<b>8</b>	<b>GCS Commands</b>	<b>129</b>
8.1	Notation.....	129
8.2	GCS Syntax for Syntax Version 2.0 .....	129
8.3	Target and Sender Address .....	132
8.4	Variables .....	132
8.5	Command Overview .....	134
8.6	Command Descriptions for GCS 2.0 .....	138
8.7	Error Codes.....	220

<b>9</b>	<b>Adapting Settings</b>	<b>245</b>
9.1	Settings of the C-663 .....	245
9.2	Changing Parameter Values in the C-663.....	245
9.2.1	General Commands for Parameters.....	246
9.2.2	Commands for Fast Access to Individual Parameters .....	247
9.2.3	Saving Parameter Values in a Text File .....	247
9.2.4	Changing Parameter Values: General Procedure .....	248
9.3	Creating or Changing a Positioner Type .....	251
9.4	Parameter Overview.....	254
<b>10</b>	<b>Maintenance</b>	<b>263</b>
10.1	Cleaning the C-663 .....	263
10.2	Updating Firmware.....	263
<b>11</b>	<b>Troubleshooting</b>	<b>269</b>
<b>12</b>	<b>Customer Service</b>	<b>273</b>
<b>13</b>	<b>Technical Data</b>	<b>275</b>
13.1	Specifications.....	275
13.1.1	Data Table.....	275
13.1.2	Maximum Ratings.....	277
13.1.3	Ambient Conditions and Classifications .....	277
13.2	Dimensions .....	278
13.3	System Requirements.....	279
13.4	Pin Assignment .....	279
13.4.1	Motor and Sensor .....	279
13.4.2	I/O .....	281
13.4.3	C-170.IO Cable for Connecting to the I/O Socket.....	281
13.4.4	Joystick.....	282
13.4.5	C-819.20Y Cable for C-819.20 Joystick .....	283
13.4.6	RS-232 In and RS-232 Out.....	283
13.4.7	Power Adapter Connector.....	285
<b>14</b>	<b>Old Equipment Disposal</b>	<b>287</b>
<b>15</b>	<b>EU Declaration of Conformity</b>	<b>289</b>

# 1 About this Document

## In this Chapter

Objective and Target Audience of this User Manual .....	1
Symbols and Typographic Conventions .....	1
Definition of Terms .....	2
Figures .....	3
Other Applicable Documents .....	4
Downloading Manuals .....	4

### 1.1 Objective and Target Audience of this User Manual

This user manual contains the information required for using the C-663 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 (p. 4) on our website.

### 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 cause damage to equipment.

➤ Precautionary measures for avoiding the risk.

#### **INFORMATION**

Information for easier handling, tricks, tips, etc.

#### **Symbol/Label**

#### **Meaning**

- |    |   |
|----|---|
| 1. | Action consisting of several steps with strict sequential order |
| 2. |   |

<b>Symbol/Label</b>	<b>Meaning</b>
➤	Action consisting of one or more steps without relevant sequential order
■	List item
p. 5	Cross-reference to page 5
<b>RS-232</b>	Labeling of an operating element on the product (example: socket of the RS-232 interface)
	Warning sign on the product referring to detailed information in this manual.
<b>Start &gt; Settings</b>	Menu path in the PC software (example: to open the menu, the <b>Start</b> and <b>Settings</b> menu items must be clicked in succession)
POS?	Command line or a command from PI's General Command Set (GCS) (example: Command to get the axis position).
<b>Device S/N</b>	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

<b>Term</b>	<b>Explanation</b>
Axis	Also referred to as "logical axis". The logical axis represents the motion of the mechanics in the firmware of the C-663. For mechanics that allow motion in several directions (e.g., in X, Y, and Z), each direction of motion corresponds to a logical axis.
Daisy chain	Wiring diagram by which one controller is connected to the next in sequence (series connection principle). Here the first controller is connected directly to the PC. The additional controllers are always connected to the ones that precede them so that a chain is formed. The signal to and from a controller goes to the PC via the previous controllers.
Dynamics profile	Comprises the target position, velocity, and acceleration of the axis calculated by the profile generator of the C-663 for any point in time of the motion. The calculated values are called "commanded values".
Firmware	Software that is installed on the controller.

Term	Explanation
Volatile memory	RAM module where the parameters are saved when the controller is switched on (working memory). The parameter values in the volatile memory determine the current behavior of the system. The parameter values in the volatile memory are also referred to as "Active Values" in the PC software from PI.
GCS	PI General Command Set; command set for PI controllers. Piezo drivers and servo controllers can be operated together with minimal programming effort thanks to GCS.
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.
Microstep resolution	A stepper motor moves the axis it is driving step by step. The path, which is covered during the changeover of the current from one winding to the next, is called full step. In the case of the C-663, a full step is divided into 2048 microsteps. Due to the high microstep resolution, on the one hand, very small motion is possible, on the other hand, smooth running is achieved.
PC software	Software installed on the PC.
Nonvolatile memory	Memory module (read-only memory, e.g., EEPROM or flash memory) where the default values of the parameters are loaded to the volatile memory when the controller is started. The parameter values in the nonvolatile memory are also referred to as "startup values" in the PC software from PI.
Positioner	Mechanics connected to the C-663. 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. In this manual, actuators, i.e., drive components without a motion platform (e.g., precision linear actuators), are designated as positioners as well.
Control value	The control value is the output value of the position controller.

## 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-663.12	MS242EK Short Instructions for Digital Motor Controllers
PI GCS driver library for use with NI LabVIEW software	SM158E Software Manual
Driver Merge Tool for use with NI LabVIEW software	SM154E Software Manual
PI MATLAB Driver GCS 2.0	SM155E Software Manual
PI GCS 2.0 DLL	SM151E Software Manual
GCS array data format description	SM146E Software Manual
PIMikroMove	SM148E Software Manual
PIStages3Editor: Software for managing the positioner database	SM156E Software Manual
PIUPdateFinder: Updating PI Software	User manual A000T0028
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

## 1.6 Downloading Manuals

### INFORMATION

If a manual is missing or problems occur with downloading:

- Contact our customer service department (p. 273).

### Downloading manuals

1. Open the website [www.pi.ws](http://www.pi.ws).
2. Search the website for the product number (e.g., P-882) or the product family (e.g., PICMA® bender).
3. Click the corresponding product to open the product detail page.
4. Click **Downloads**.

The manuals are shown under **Documentation**.

5. Click the desired manual and fill out the inquiry form.

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



## 2 Safety

### In this Chapter

Intended Use.....	7
General Safety Instructions.....	7
Organizational Measures .....	8

#### 2.1 Intended Use

The C-663 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-663 is intended for operating PI positioners equipped with stepper motors.

The C-663 is intended both for open-loop operation and for position-controlled operation of 2-phase stepper motors. Position-controlled operating requires incremental sensors. In addition, the C-663 can read and process the reference point and limit switch signals from the connected positioner.

The C-663 may only be used in compliance with the technical specifications and instructions in this user manual. The user is responsible for process validation.

#### 2.2 General Safety Instructions

The C-663 is built according to state-of-the-art technology and recognized safety standards. Improper use can result in personal injury and/or damage to the C-663.

- Use the C-663 for its intended purpose only, and only when it is in perfect technical condition.
- Read the user manual.
- Eliminate any malfunctions that may affect safety immediately.

The operator is responsible for the correct installation and operation of the C-663.

- Install the C-663 near the power supply so that the power plug can be quickly and easily disconnected from the mains.
- Use the components supplied (power supply and power cord) to connect the C-663 to the power supply.

- If one of the components supplied for connecting to the power supply has to be replaced, use a sufficiently rated component.

## 2.3 Organizational Measures

### User manual

- Keep this user manual with the C-663 always.  
The latest versions of the user manuals are available for download (p. 4) on our website.
- Add all manufacturer information such as supplements or technical notes to the user manual.
- If you give the C-663 to other users, also include this user manual as well as other relevant information provided by the manufacturer.
- Always work according to the complete user manual. The equipment can be damaged if your user manual is incomplete and is therefore missing important information.
- Install and operate the C-663 only after you have read and understood this user manual.

### Personnel qualification

The C-663 may only be installed, started up, operated, maintained, and cleaned by authorized and appropriately qualified personnel.

## 3 Product Description

### In this Chapter

Product View.....	9
Scope of Delivery .....	12
Accessories.....	12
Overview of PC Software .....	13
Positioner Databases .....	15
Communication Interfaces.....	16
Functional Principles.....	17

### 3.1 Product View

#### 3.1.1 Front Panel



Figure 1: Front panel of the C-663.12

Labeling	Type	Function
	Mini-USB type B	Universal serial bus for connection to the PC; do not connect if <b>RS-232 In</b> is already connected.
<b>RS-232 In</b>	D-sub 9 (m) (p. 283)	Serial connection to the PC or to the previous controller in a daisy chain network; do not connect to the PC if the USB interface is already connected.
<b>RS-232 Out</b>	D-sub 9 (f) (p. 283)	Serial connection to the subsequent controller in a daisy chain network

Labeling	Type	Function
<b>STA</b>	LED green	Controller state: <ul style="list-style-type: none"> <li>▪ Lights up continuously: C-663 is ready for normal operation</li> <li>▪ Flashing: C-663 is in firmware update mode</li> <li>▪ Off: C-663 is not connected to the supply voltage</li> </ul>
<b>ERR</b>	LED red	Error indicator: <ul style="list-style-type: none"> <li>▪ On: Error (error code <math>\neq 0</math>)</li> <li>▪ Off: No error (error code = 0)</li> </ul> <p>The error code can be queried with the <code>ERR?</code> command. The query resets the error code to zero and the LED is switched off.</p>
<b>Mode Baud Addr</b>	8-bit DIP switch (p. 60, p. 60)	Setting of <ul style="list-style-type: none"> <li>▪ Device address (<b>Addr</b>)</li> <li>▪ Baud rate for communication with the PC (<b>Baud</b>)</li> </ul> <p>The switches 7 and 8 (<b>Mode</b>) have no function.</p>

### 3.1.2 Rear Panel



Figure 2: Rear panel of the C-663.12

Labeling	Type	Function
<b>12 to 48 VDC Max. 2 A</b>	DC power socket (Kycon), 4-pole (f), lockable (p. 285)	Connector for the supply voltage

<b>Labeling</b>	<b>Type</b>	<b>Function</b>
<b>I/O</b>	Mini-DIN, 9-pole (f) (p. 281)	Digital inputs/outputs: <ul style="list-style-type: none"> <li>▪ Outputs: Trigger external devices</li> <li>▪ Inputs: Use in macros or as switch signals</li> </ul> Analog inputs: <ul style="list-style-type: none"> <li>▪ Use in macros or for scanning processes</li> </ul>
<b>Joystick</b>	Mini-DIN, 6-pole (f)	Connector for analog joystick <ul style="list-style-type: none"> <li>▪ Inputs for signals from the joystick axes and buttons</li> <li>▪ Output for the supply voltage of the joystick</li> </ul>
<b>Motor</b>	HD D-sub 26 (f) (p. 279)	Positioner's connector <ul style="list-style-type: none"> <li>▪ Output for motor phases</li> <li>▪ Input of the signals of the incremental position sensor</li> <li>▪ Signal input from the limit switches and reference switch</li> </ul>
	Screw and toothed washer Grounding the C-663 (p. 50)	Ground connection If potential equalization is required, the screw can be connected to the grounding system.

### 3.1.3 Type Plate

<b>Labeling</b>	<b>Function</b>
	Data matrix code (example; contains the serial number)
<b>C-663.12</b>	Product name
<b>PI</b>	Manufacturer's logo
<b>116056789</b>	Serial number (example), individual for each C-663 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. 287)
	CE conformity mark
WWW.PI.WS	Manufacturer's address (website)

## 3.2 Scope of Delivery

Item	Component
C-663.12	Controller
C-501.48120DIN4	Wide input range power supply 48 V 120 W, mini-DIN, 4-pin
3763	Power cord
000036360	USB cable (type A to Mini B) for connecting to the PC
C-815.34	RS-232 null modem cable, 3 m, 9/9-pin
C-862.CN	Network cable for daisy chain network, 30 cm
C-815.AA42	Plug adapter for D-sub 15 (f) to HD D-sub 26 (m)
C-990.CD1	Data storage medium with PC software from PI
MS242EK	Short instructions for digital motor controllers

## 3.3 Accessories

Article	Component
C-815.LSM2	Motor cable, stepper motor, HD D-sub 26 (f) on HD D-sub 26 (m), 3 m
C-815.AC32	Converter for sin/cos sensor signals on AB quadrature signals (RS-422), HD D-sub 26 (m)
C-862.CN2	Network cable for daisy chain network, 3 m
C-819.20	Analog joystick for 2 axes, see "Joysticks Available" (p. 108) for details
C-819.20Y	Y cable for connecting 2 controllers to C-819.20 joystick
C-819.30	Analog joystick for 3 axes, see "Joysticks Available" (p. 109) for details
C-170.PB	Pushbutton box with 4 buttons and 4 LEDs
C-170.IO	I/O cable, 2 m, open end (p. 281)

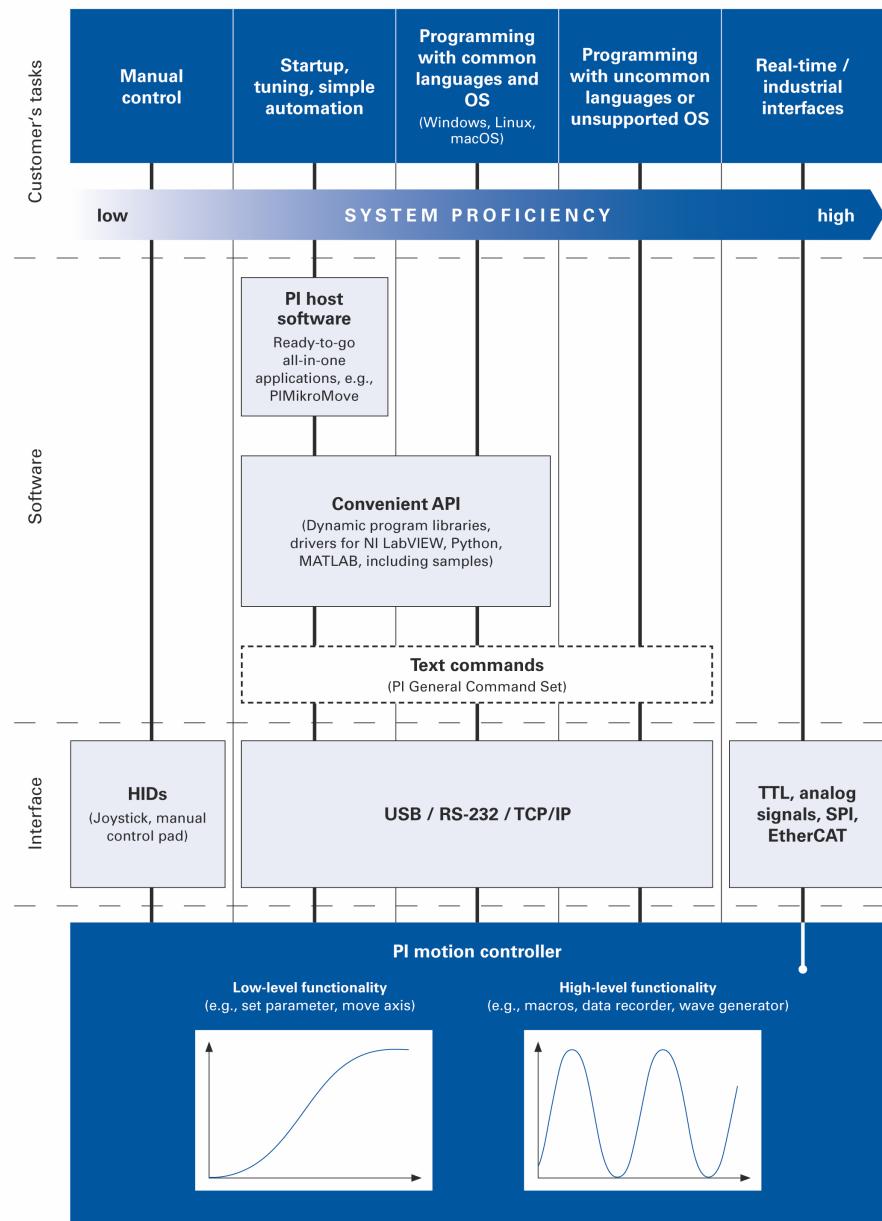


Connection to the **I/O** socket of the C-663, sends 4 TTL input signals and displays the state of the 4 digital outputs via the LEDs.

To order, contact our customer service department (p. 273).

## 3.4 Overview of PC Software

Basically, PI systems can be controlled as follows:



The following table shows the PC software on the PI software CD. The specified operating systems stand for the following versions:

- Windows: Versions 8.1, 10 (32-bit, 64-bit)
- Linux: Kernel 2.6, GTK 2.0, from glibc 2.15

<b>PC software</b>	<b>Operating system</b>	<b>Short description</b>	<b>Recommended use</b>
Dynamic program library for GCS	Windows, Linux	Allows software programming for the C-663 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 if communication is to be established via USB (with Linux only via virtual COM port) or a daisy chain network.
NI LabVIEW drivers	Windows, Linux	NI LabVIEW is a software for data acquisition and process control (must be ordered separately from National Instruments). The C-663 NI LabVIEW software of PI is a collection of virtual instrument drivers (VI drivers) for the C-663 controller. These drivers support the GCS.	For users who want to use NI LabVIEW to program their application.
NI LabVIEW Merge Tool	Windows	The NI LabVIEW Merge Tool allows you to combine product-specific NI LabVIEW drivers from PI with each other.	For users who want to operate several products from PI at the same time while using NI LabVIEW.
MATLAB drivers	Windows	MATLAB is a development environment and programming language for numerical calculations (must be ordered separately from MathWorks). The PI MATLAB driver consists of a MATLAB class that can be included in any MATLAB script. This class supports the PI General Command Set.	For users who want to use MATLAB to program their application. Software from MathWorks is to be obtained separately. The PI MATLAB driver does not require any additional MATLAB toolboxes.
PIMikroMove	Windows	Graphical user interface for Windows, which can be used for controllers from PI: <ul style="list-style-type: none"> <li>▪ Start the system without programming effort</li> <li>▪ Graphic representation of the motion</li> </ul>	For users who want to do simple automation tasks or test their equipment before or instead of programming an application. No command knowledge is necessary to operate PIMikroMove. A log window showing the

<b>PC software</b>	<b>Operating system</b>	<b>Short description</b>	<b>Recommended use</b>
		<ul style="list-style-type: none"> <li>▪ Macro functionality for storing command sequences on the PC (host macros)</li> <li>▪ Complete environment for command entry</li> </ul> <p>PIMikroMove uses the dynamic program library to supply commands to the controller.</p>	commands sent makes it possible to learn how to use the commands.
PITerminal	Windows, Linux	Simple user interface that can be used for nearly all PI controllers.	For users who want to send GCS commands directly to the controller.
PIStages3Editor	Windows	Program for opening and editing positioner databases in .db format.	For users who want to deal intensively with the contents of positioner databases.
PI Update Finder	Windows	Checks the PI software installed on the PC. If newer versions of the PC software are available on the PI server, they are offered for download.	For users who want to update the PC software.
PI Firmware Updater	Windows	Program for user support when updating the firmware of the C-663.	For users who want to update the firmware.
USB driver	Windows	Driver for the USB interface	For users who want to connect the controller to the PC via the USB interface.

### 3.5 Positioner Databases

You can select a parameter set appropriate 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.

<b>File name</b>	<b>Description</b>
PISTAGES3.DB	Delivery includes parameter sets for all standard positioners from PI and PI miCos and is automatically saved to the PC when the PC software is installed New parameter sets can be created, edited, and saved (p. 251).
<Produkt>.db e.g.: M-xxxxxx.db	Includes the parameter set for a custom positioner. In order for the parameter set to be selected in the PC software, it must be added to the PISTAGES3.DB first, see "Installing Custom Positioner Databases" (p. 48).

Parameters loaded from a positioner database are marked in color in the parameter overview (p. 254).

For more information on the positioner database, see the manuals for the PIStages3Editor and the PI GCS program library.

### **INFORMATION**

If the pistages2.dat and pimicosstages2.dat positioner databases are on your PC:  
Positioner databases in .dat format are only installed for compatibility reasons and **not** used for the C-663 described in this manual.

## **3.6 Communication Interfaces**

### **Communication interfaces available**

The C-663 can be controlled with ASCII commands from a PC: The connection to the PC can be made via a direct connection or via a daisy-chain network. The following interfaces of the C-663 can be used for direct connection to the PC:

- Serial RS-232 connection
- USB connection

Only one of the two interfaces may be connected to the PC at all times.

### **Default communication settings**

Interface	Property	Default value
RS-232	Baud rate	115200 Settings for DIP switches 5 and 6; see "Baud Rate" (p. 62) Other: 8 data bits and 1 stop bit, without parity; internal buffers do not require a handshake

### **Daisy chain network**

Using a daisy chain network, up to 16 controllers can be connected to the PC via a single RS-232 or USB connection. Interlinking occurs in series. See also "Definition of Terms" (p. 2).

## 3.7 Functional Principles

### 3.7.1 Block Diagram

The C-663 controls the motion of the logical axis of a positioner. The following block diagram shows how the C-663 generates the output signal for the axis connected:

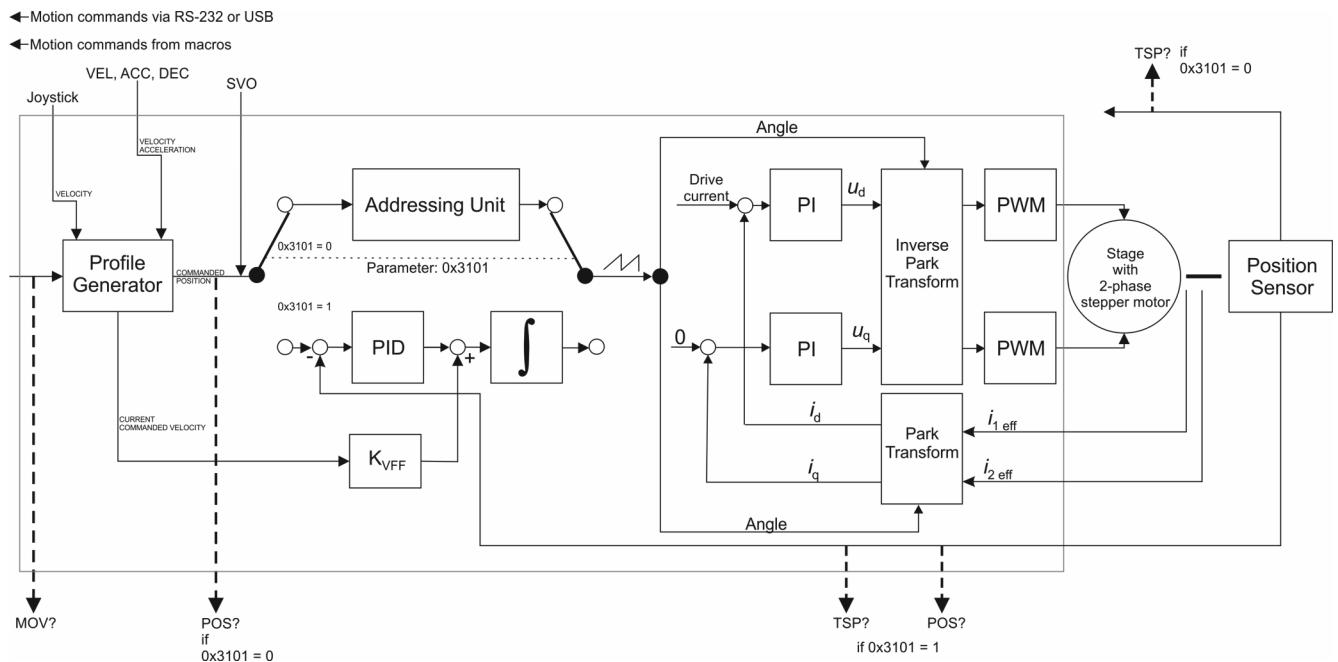


Figure 3: C-663.12: Generation of the output current

The C-663.12 supports positioners without sensor, and positioners equipped with an incremental position sensor.

### 3.7.2 Commandable Elements

The following table contains the elements that can be commanded with GCS commands (p. 138).

Element	Num ber	ID	Description
Logical axis	1	1 (modifi able)	<p>The logical axis represents the motion of the positioner in the firmware of the C-663. It corresponds to an axis of a linear coordinate system.</p> <p>All commands for the motion of a positioner refer to logical axes.</p> <p>Motion for logical axes is commanded in the firmware of the C-663 (i.e., for the directions of motion of a positioner).</p> <p>The axis identifier can be queried with the <code>SAI?</code> command and modified with the <code>SAI</code> command. It can consist up to 8 characters; valid characters are 1234567890ABCDEFGHIJKLMNPQRSTUVWXYZ-_.</p> <p>The new axis identifier is only transferred to the volatile memory of the C-663. A changed axis identifier can be stored permanently in the C-663 with the <code>WPA</code> command (p. 219).</p>
Input signal channel	1	1	<p>Channel for an incremental sensor; input via the <b>Motor</b> (p. 279) socket</p> <p>The sensor position can be queried with the <code>TSP?</code> command and set with the <code>TSP</code> command.</p> <p>For further information, see "Operating modes" (p. 20), "Physical Units" (p. 22), and "Referencing" (p. 38).</p>
Analog inputs	6	1 to 6	<p>The analog input lines with the identifiers 1 to 4 are the inputs 1 to 4 of the <b>I/O</b> (p. 281) socket. Their number is displayed with the <code>TAC?</code> command and their values can be queried with the <code>TAV?</code> command. Note that these lines can also be used as digital inputs (see below).</p> <p>Additional analog input lines are located at the <b>Joystick</b> socket.</p> <p>These lines are not output via <code>TAC?</code> and <code>TAV?</code>.</p> <p>The values of all inputs can be recorded via record option 81 of the <code>DRC</code> command.</p>
Digital outputs	4	1 to 4	<p>1 to 4 identify digital output lines 1 to 4 of the <b>I/O</b> (p. 281) socket.</p> <p>For further information, see "Digital Output Signals" (p. 89).</p>

<b>Element</b>	<b>Num ber</b>	<b>ID</b>	<b>Description</b>
Digital inputs	4	1 to 4	1 to 4 identify digital input lines 1 to 4 of the <b>I/O</b> socket (p. 281), which can also be used as analog inputs (see above). For further information, see "Digital Input Signals" (p. 96).
Joystick	1	1	An analog joystick can be connected to the <b>Joystick</b> socket.
Joystick axis	1	1	Pin 4 of the <b>Joystick</b> socket: Axis 1 of joystick 1
Joystick button	1	1	Pin 6 of the <b>Joystick</b> socket: Button 1 of joystick 1  For further information, see "Joystick Control" (p. 102). For data recorder configuration with the <b>DRC</b> command, the following data source identifiers apply: 5 = Axis 1 of joystick 1 6 = Button 1 of joystick 1
Data recorder tables	4	1 to 4	The C-663 has 4 data recorder tables (query with <b>TNR?</b> ) with 1024 data points per table.
Controller address	1	1 to 16	The controller address can be set in the range from 1 to 16 with the DIP switches on the front panel of the C-663. In a daisy chain (p. 53), each controller must have a unique address (p. 132).
Overall system	1	1	C-663 as an overall system

### 3.7.3 Important Components of the Firmware

The firmware of the C-663 provides the following functional units:

<b>Firmware component</b>	<b>Description</b>
ASCII commands (GCS)	<p>Communication with the C-663 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-663</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. 134).</p>

Firmware component	Description
Parameters	Parameters reflect the properties of the positioner connected (e.g., travel range) and specify the behavior of the C-663 (e.g., selecting the operating mode, settings for the servo algorithm). Parameter values can be changed to adapt the system to the particular application. For further information, see "Adapting Settings" (p. 245). Write access for the parameters of the C-663 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.
Profile Generator	The profile generator calculates the dynamics profile to determine the target position ("commanded position"), velocity, and acceleration of the axis at each time of the movement. For further information, see "Generating a Dynamics Profile" (p. 25).
Servo algorithm	Closed-loop operation: The position error resulting from the difference between the commanded position (output of the profile generator) and the actual position (sensor feedback) runs through a PID servo algorithm. For further information, see "Position Control" (p. 28).
Data Recorder	The C-663 contains a real-time data recorder (p. 87). The data recorder can record various signals (e.g., position, analog input) from different data sources (e.g., logical axis or input channels).
Macros	The C-663 can save macros (p. 110). 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-663 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. 110).

The firmware can be updated with a tool (p. 263).

### 3.7.4 Operating Modes

The C-663 supports the following operating modes:

- Operation with position control: "Closed-loop operation"
- Operation without position control: "Open-loop operation"

The designations "closed-loop operation" and "open-loop operation" in this manual refer only to the position control and **not** to the current regulation (p. 31) of the C-663. The current regulation is switched on during operation with position control as well as in operation without position control.

The operating mode of the C-663 is determined by the value of the parameter ***Stepper Closed-Loop Operation*** (0x3101):

Parameters	Description and Possible Values
<b><i>Stepper Closed-Loop Operation</i></b> 0x3101	Operating mode of the C-663 0 = Open-loop operation 1 = Closed-loop operation The value of the parameter can only be changed if the motor is deactivated. See "Setting the Operating Mode" (p. 82) for details.

### INFORMATION

Motion of the axis is possible both in open-loop and in closed-loop operation, if the motor is activated with the SVO command (p. 207).

The current activation state of the motor can be queried with the SVO? command (p. 208).

Differences between the operating modes:

	<b>Open-Loop Operation</b>	<b>Closed-Loop Operation</b>
Correction of position errors	Position errors are <b>not</b> corrected.	Position errors are corrected by a servo algorithm. For further information see "Position Control" (p. 28).
Use of a position sensor	Optional. Used only to display the position.	Required. Used for position control.
Get the current axis position	Using the POS? command. The answer to POS? is based on the commanded position (output of the profile generator).  If a position sensor is present: The TSP? command queries the position measured.  For the adjustment of the values queried with POS? and TSP?, see "Referencing" (p. 38).	Using the POS? command. The answer to POS? is based on the position measured by the sensor.

	<b>Open-Loop Operation</b>	<b>Closed-Loop Operation</b>
Determination of the length unit for position data For further information see "Physical units" (p. 22).	Is determined by the factor for microsteps per physical unit of length (parameter 0xE, 0xF). Exception: The length unit for the position measured by the sensor is determined by the factor for counts per physical unit of length (parameter 0x9A, 0x9B).	Is determined by the factor for counts per physical unit of length (parameter 0x9A, 0x9B).

### 3.7.5 Physical Units

The C-663 internally uses the following length units for position data:

- Microsteps of the motor
- Count pulses of the sensor

The conversion of an internal length unit into the desired physical length unit for the command is done with a factor. 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 microsteps per physical unit of length determines the length unit for motion commands and for positional queries with POS? in open-loop operation.

<b>Parameters</b>	<b>Description and Possible Values</b>
<b>Numerator Of The Counts-Per-Physical-Unit Factor</b> 0xE	Numerator and denominator of the factor for microsteps per physical length unit 1 to 1,000,000,000 for each parameter The numerator and denominator of the factor can be calculated from the technical data of the positioner: Numerator = number of full steps / U * 2048 Denominator = Drive screw pitch in phys. length unit / U
<b>Denominator Of The Counts-Per-Physical-Unit Factor</b> 0xF	

The factor for the counts per physical unit of length determines the length unit for motion commands and for positional queries with POS? in closed-loop operation. In open-loop operation, the factor determines the length unit of the position measured by the sensor (TSP / TSP?).

Parameters	Description and Possible Values
<b>Numerator Of The Counts-Per-Physical-Unit Factor For Sensor</b> 0x9A	Numerator and denominator of the factor for counts per physical length unit 1 to 1,000,000,000 for each parameter The numerator and denominator of the factor can be calculated from the resolution of the encoder scale used.
<b>Denominator Of The Counts-Per-Physical-Unit Factor For Sensor</b> 0x9B	The factor has no influence on the stability of the servo loop, but is used for the input and output scaling of position values.

### INFORMATION

The factors for the conversion of microsteps and counts into the desired physical unit of length determine the values of all other length-based parameters of the C-663. When adapting the C-663's parameters to the connected positioner, parameters 0xE, 0xF, 0x9A, and 0x9B must therefore be set before all other parameters.

If you load the parameters from a positioner database (e.g., when working with PIMikroMove), the correct sequence is maintained automatically.

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. The unit is for example, "mm" if the parameters 0xE, 0xF, 0x9A, and 0x9B are set so that microsteps and counts are converted in millimeters. The unit for rotation stages is normally "deg". The value of the parameter 0x07000601 is not evaluated by the C-663 but is used by the PC software for display purposes. Examples: 1 Microstep = 100 nm Number of microsteps per physical unit of length: 10000:1 → Unit: mm 1 Microstep = 0.254 mm Number of microsteps per physical unit of length: 100:1 → Unit: Inch

### 3.7.6 Motion Triggering

#### **INFORMATION**

Motion of the axis is possible both in open-loop and in closed-loop operation, if the motor is activated with the SVO command (p. 207).

The current activation state of the motor can be queried with the SVO? command (p. 208).

The following table is valid for motion in open-loop and 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 doing 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. 110).
Joystick control	JON	Activates or deactivates a joystick connected to the controller.
	JAX	Specifies the axis that is controlled by a joystick connected to the controller. The joystick controls the speed of the axis. Motion commands are not allowed when the joystick control is activated for the axis.
		For further commands see "Commands and Parameters for Joystick Control" (p. 103).

#### **INFORMATION**

Absolute target positions can only be commanded if the axis was referenced beforehand; see "Referencing" (p. 38).

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 Generating a Dynamics Profile

The profile generator is used in both open-loop and closed-loop operation.

The profile generator does 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-663 depends on the motion parameters which are specified by commands (p. 138), parameters and/or by joystick. The following applies to both open-loop and closed-loop operation.

Motion parameter	Commands	Parameters	Remarks
Acceleration (A)	ACC ACC?	Acceleration (Parameter 0xB; physical length units/s <sup>2</sup> ); change with the ACC or with SPA / SEP command; can be stored with WPA.	Is limited by parameter 0x4A.
Deceleration (D)	DEC DEC?	Deceleration (Parameter 0xC; physical unit of length/s <sup>2</sup> ); change with the DEC or with SPA / SEP command; can be saved with WPA.	Is limited by parameter 0x4B.
Velocity (V)	VEL VEL?	Velocity (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. When a joystick is connected to the C-663 and the joystick is activated by the JON command, a factor is applied to the current velocity set with the VEL command. For further information, see "Joystick Control" (p. 102).

Motion parameter	Commands	Parameters	Remarks
Target position at the end of the motion	MOV MVR GOH STE	-	<p>When a joystick is connected to the C-663 and the joystick is activated by the JON command, the soft limits are set as respective target position. When deactivating the joystick, the target position is set to the current position for the joystick-controlled axes. For further information, see "Joystick Control" (p. 102).</p> <p>When activating the motor with the SVO command or when stopping the axis motion with the #24, STP or HLT commands, the target position is set to the current position.</p>

The profile generator of the C-663 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.

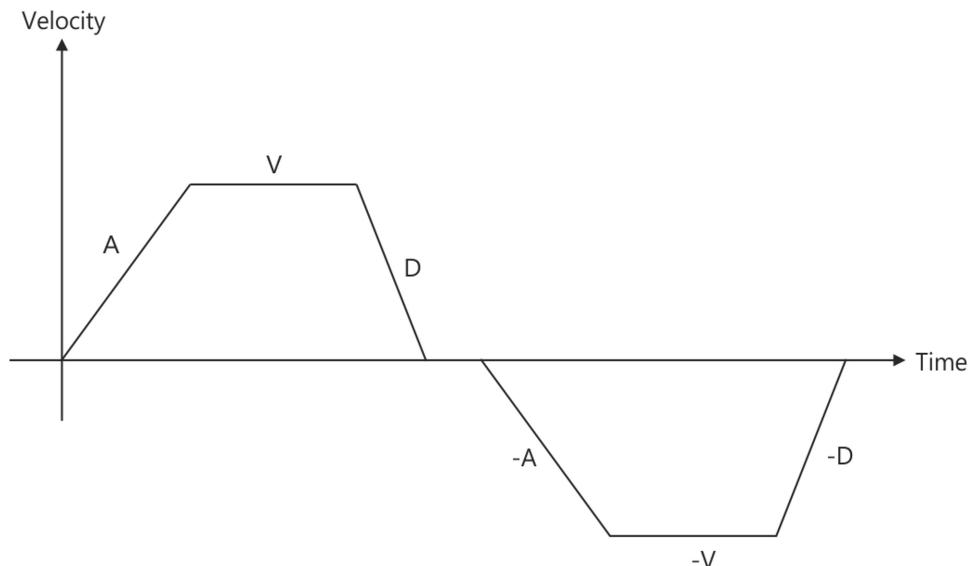


Figure 4: Basic trapezoidal velocity profile; A = acceleration, D = deceleration, V = velocity

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.

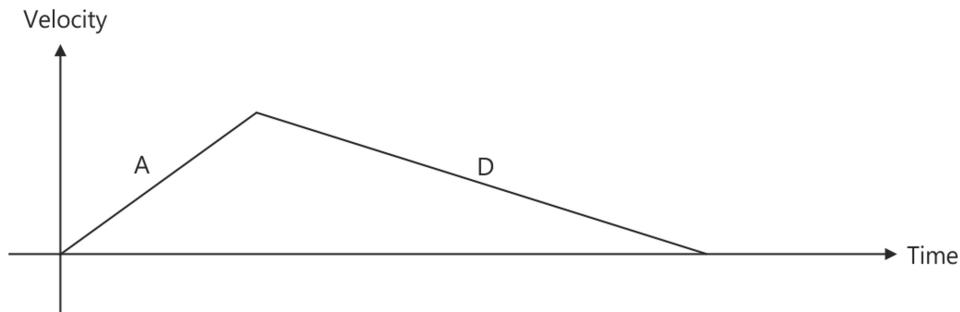


Figure 5: Basic trapezoidal velocity profile; A = acceleration, D = deceleration, no constant velocity

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.

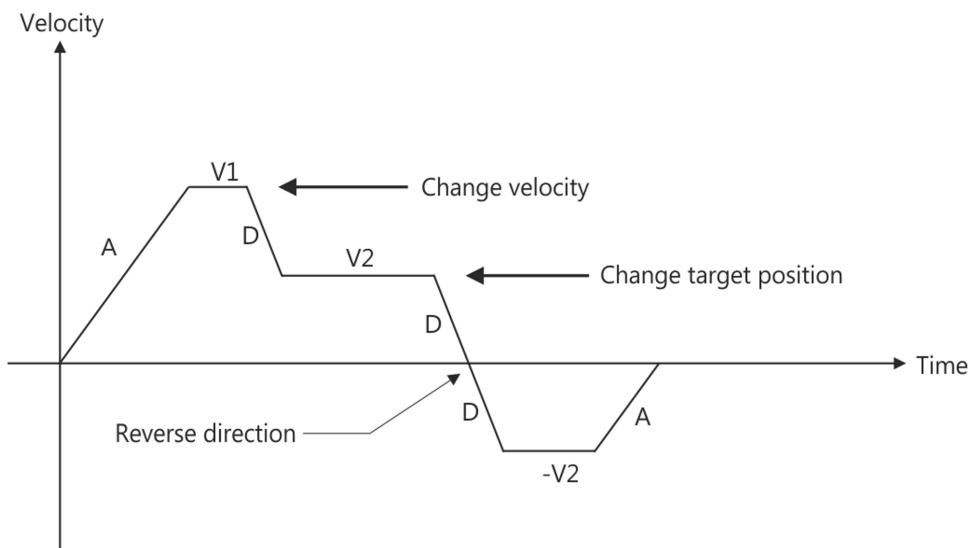


Figure 6: Complex trapezoidal profile with parameter changes; A = acceleration; D = deceleration; V1, V2, -V2 = velocities

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 Position Control

In closed-loop operation, the following corrections optimize the control value for both motor phases and therefore the settling behavior of the system:

- P-I-D servo algorithm (proportional-integral-differential)
- Velocity feedforward
- Limitation of the start / stop step frequency of the motor

The C-663 uses two groups of servo control parameters: The settings of parameter group 1 (motion) are evaluated during axis motion and the settings of parameter group 2 (end position) are evaluated at the target position. Switching from parameter group 1 to parameter group 2 is done via a configurable velocity threshold value.

Each parameter group contains the following settings:

- P, I, D terms and I limit for the servo algorithm

#### P-I-D servo algorithm

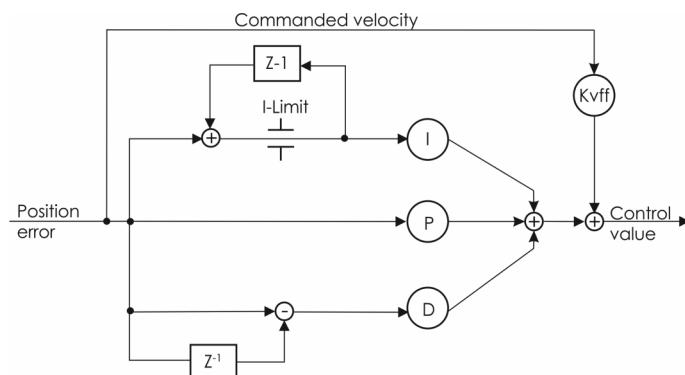


Figure 7: PID algorithm with feed-forward velocity control

The position error that results from the difference between the commanded position (output of the profile generator) and the actual position (sensor feedback) runs through the PID 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. 76).

Parameters	Description and Possible Values
<b>P-Term 1 (Motion)</b> 0x3111	Proportional constant 1 (dimensionless) 0 to 32767 Objective: Fast correction of the position error during the axis motion
<b>I-Term 1 (Motion)</b> 0x3112	Integration constant 1 (dimensionless) 0 to 32767 Objective: Reduction of the static position error during axis motion

<b>Parameters</b>	<b>Description and Possible Values</b>
<b>D-Term 1 (Motion)</b> 0x3113	Differential constant 1 (dimensionless) 0 to 32767 Default setting: 0 Objective: Damping of fast control oscillation during axis motion The default setting of the parameter should <b>not</b> be changed.
<b>I-Limit 1 (Motion)</b> 0x3114	Limit of the integration constant 1 (dimensionless) 0 to 32767 Effective during axis motion
<b>P-Term 2 (End Position)</b> 0x3121	Proportional constant 2 (dimensionless) 0 to 32767 Objective: Fast correction of the position error at the target position
<b>I-Term 2 (End Position)</b> 0x3122	Integration constant 2 (dimensionless) 0 to 32767 Objective: Reduction of the static position error at the target position
<b>D-Term 2 (End Position)</b> 0x3123	Differential constant 2 (dimensionless) 0 to 32767 Default setting: 0 Objective: Damping of rapid control oscillation at the target position The default setting of the parameter should <b>not</b> be changed.
<b>I-Limit 2 (End Position)</b> 0x3124	Limit of the integration constant 2 (dimensionless) 0 to 32767 Effective with the target position
<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.

The input of the servo algorithm can be configured for the C-663 with the following parameters:

<b>Parameters</b>	<b>Description and Possible Values</b>
<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 input factor of the servo loop scales the servo control

Parameters	Description and Possible Values
<b>Denominator Of The Servo-Loop Input Factor</b> 0x5B	parameters (P, I, D term) and decouples them from the encoder resolution. The scaling of the servo control parameters is particularly important with very high encoder resolution. Example: An input factor 10 scales down the servo loop 10: 1. The input factor of the servo loop is independent of the factors for the microsteps per physical length unit (0xE and 0xF) and the counts per physical length unit (0x9A, 0x9B). Numerator and denominator of the servo-loop input factor should not be changed.

### Velocity feedforward

The feedforward velocity control can be set via the following parameters:

Parameters	Description and Possible Values
<b>Kvff (motion)</b> 0x3117	Weighting the velocity as feedforward signal 0 to 65535 as floating point value The weighting factor is multiplied by the commanded velocity (profile generator output) during axis motion. Target: Minimizing the position error during axis motion

### Switching from parameter group 1 to parameter group 2

Switching of the servo control parameters is set via the following parameter:

Parameters	Description and Possible Values
<b>Velocity Threshold For Switching To End-Position PID Set (Phys. Unit/s)</b> 0x3116	Velocity threshold value for switching the servo control parameters Going below this velocity at the end of axis motion switches from parameter group 1 (motion) to parameter group 2 (end position)

### Limitation of the start / stop step frequency

The start / stop step frequency of the motor in closed-loop operation can be limited by the following parameter:

Parameters	Description and Possible Values
<b>Stepper Start/Stop Frequency (full steps/s)</b> 0x3102	Maximum full-step frequency when starting and stopping the motion This parameter is only evaluated in closed-loop operation.

### 3.7.9 Motor Control

The C-663 has the following functions for motor control:

- **Current regulation:** The current control determines the currents for the two motor phases.
- **Holding current:** In order to avoid the heating of the motor during standstill, the C-663 switches from the operating current to the holding current after a delay time.

#### INFORMATION

The current regulation of the C-663 is switched on during operation with position control as well as in operation without position control. The current control is switched on when the motor is activated with the SVO (p. 207) command.

The current activation state of the motor can be queried with the SVO? (p. 208) command.

The motor control can be configured with the following parameters:

Parameters	Description and Possible Values
<b>Holding Current</b> 0x40	Holding current When the axis reaches the target position, the C-663 switches from the operating current to the holding current after the holding current delay (parameter 0x42). To avoid heating the motor, the value for the holding current should be less than the value for the minimum operating current (parameter 0x41).
<b>Drive Current</b> 0x41	Minimum value of the operating current The motor is supplied with operating current while the axis is in motion.
<b>Holding Current Delay (ms)</b> 0x42	Holding current delay time Delay time for switching from the operating current (parameter 0x41) to the holding current (parameter 0x40). The delay time begins when the axis reaches the target position.
<b>P-Term Current Control</b> 0x3040	Proportional constant (dimensionless) of the current regulation 0 to 2.5
<b>I-Term Current Control (Tn)</b> 0x3041	Integration constant (dimensionless) of the current regulation 0 to 2 The larger the value, the less is integrated.
<b>Current Full-Step Frequency (full steps/s)</b> 0x3106	Current full-step frequency of the motor This parameter is write-protected and can only be read.

**INFORMATION**

The parameters for the motor control can be adjusted as follows:

- For PI positioners: Load the parameter set suitable for your positioner from the positioner database, see "Starting Motion" (p. 71). **No** further adjustment of the loaded parameter values is necessary.
- For third-party positioners: Monitor the phase currents appropriately, e.g., with a multimeter while you are setting the parameters for motor control.

### 3.7.10 On-Target State

Reaching of the target position can be checked by means of the on-target status:

- 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-663 determines the on-target state on the basis of the following criteria:

- Settling window around the target position (parameter 0x36)
- Delay time for setting the on-target state (parameter 0x3F)

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

- Closed-loop operation:
  - 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 commanded position (output of the profile generator) is equal to the target position.
- Open-loop operation:
  - The commanded position (output of the profile generator) is equal to the target position.

Parameters	Description and Possible Values
<b>Settling Time (s)</b> 0x3F	Delay time for setting the on-target state 0 to 1.000 s This parameter is only evaluated in closed-loop operation.
<b>Settling Window (encoder counts)</b> 0x36	Settling window around the target position 0 to $2^{31}$ counts of the encoder Specifies the window limits. If the current position exits the settling window, the target position is no longer considered as reached. The parameter value corresponds to half the width of the window. It can only be changed if the motor is deactivated. This parameter is only evaluated in closed-loop operation.

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

The on-target state of the selected axis is output at the selected trigger output in *On Target* trigger mode (p. 92).

### 3.7.11 Reference Switch Detection

The C-663 receives the signal from the reference switch at the **Motor** socket.

Reference switch detection by the C-663 can be configured with the following parameters:

Parameters	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 (p. 99).
<b>Has Reference?</b> 0x14	Does the positioner have a reference switch? 0 = Reference switch not installed 1 = Reference switch available (signal input on motor connector)  This parameter activates or deactivates reference moves to the installed reference switch.
<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.

The signal from the reference switch of the positioner can be used for reference moves. After a reference move to the reference switch, the controller knows the absolute axis position; see "Referencing" (p. 38).

### 3.7.12 Limit Switch Detection

The C-663 receives limit switch signals at the **Motor** socket:

- Pin 12: Positive limit switch
- Pin 11: Negative limit switch

The following parameters can be used to configure how the C-663 detects the limit switches:

Parameters	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 on motor connector) 1 = Positioner does not have limit switches This parameter activates or deactivates a stop of the motion 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 motion in front of 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 hit the hard stop anyway. To prevent this, soft limits (p. 34) can be set via parameters of the C-663.

The limit switch signals can also be used for reference moves. After a reference move to a limit switch, the controller knows the absolute axis position; see "Referencing" (p. 38).

### 3.7.13 Travel Range and Soft Limits

The following parameters of the C-663 reflect the physical travel range of the positioner and define soft limits:

Parameter	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.

Parameter	Description and Possible Values
<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 done a reference move to the reference switch. The parameter value is also used for calculating the position values set after reference moves to the limit switches; this also applies when the mechanics do 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 done 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 done 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 a 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 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 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-663 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 and open-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**):
  - For logical reasons, the values are outside the soft limits specified via 0x15 and 0x30.
  - Apply both in closed-loop and open-loop operation.
  - Motion is 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. 38), 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

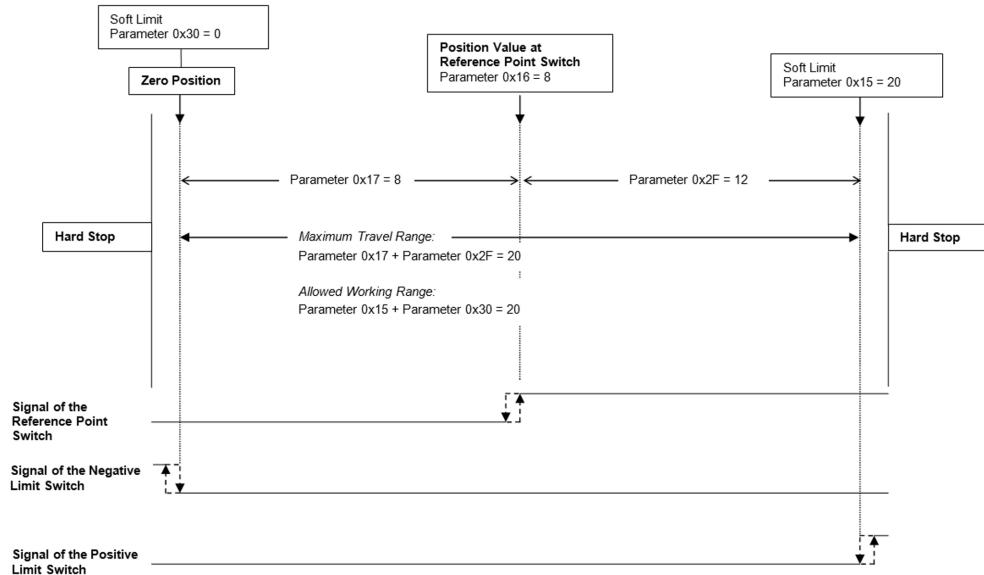


Figure 8: The travel range of the axis is not limited by soft limits.

After a reference move of the axis to the reference switch (`FRE` 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.

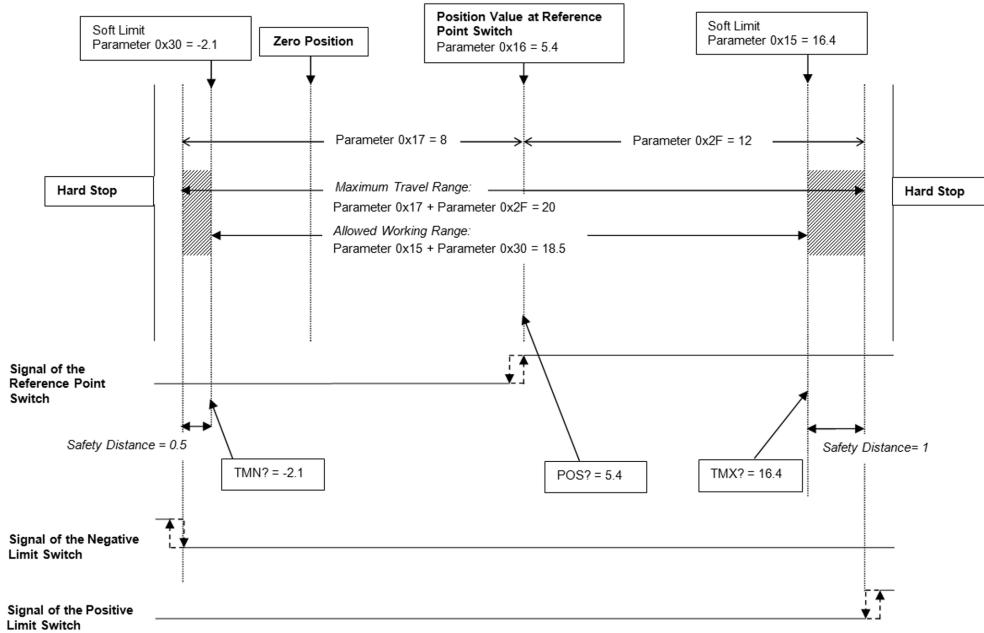


Figure 9: The travel range of the axis is limited by soft limits.

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

### 3.7.14 Referencing

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, referencing must be done for the axis.

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 selected with the `RON` command (p. 196), you can set the current position of the axis to an arbitrary value at an arbitrary point using `POS` (p. 194). Here the axis is not moved. The controller knows the absolute axis position afterwards.

**INFORMATION**

During startup using PIMikroMove, referencing is done via a reference move by default.

**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>)**)

**INFORMATION**

If there is a position sensor and **referencing is done in open-loop operation**:

Referencing only sets the current position to a defined value, which is queried by the POS? command. The position measured by the sensor is **not** set to a defined value by referencing.

If the position values queried with POS? and TSP? are to be comparable:

1. Directly after referencing, get the axis position by sending `POS?`.
2. Deactivate the motor by sending `SVO <AxisID> 0`, whereby `<AxisID>` identifies the axis.
3. Send `TSP 1 <Position>` to set the position measured by the sensor to the position value that was queried with POS?. `<Position>` denotes the position value to be set.

4. Activate the motor by sending `SVO <AxisID> 1`, whereby `<AxisID>` identifies the axis.
  5. Send `POS?` and `TSP?` consecutively to retrieve the current axis position and the sensor position. The positional values should be about the same.
- 

## Commands

The following commands are available for referencing:

Com-mand	Syntax	Function
<code>RON</code>	<code>RON {&lt;AxisID&gt; &lt;ReferenceOn&gt;}</code>	Selects the referencing method ( <code>&lt;ReferenceOn&gt;</code> ) for an axis: <ul style="list-style-type: none"> <li>▪ 0: An absolute position value can be assigned with <code>POS</code> or <code>TSP</code>, or a reference move can be started with <code>FRF</code>, <code>FNL</code> or <code>FPL</code>.</li> <li>▪ 1 (default): A reference move must be started with <code>FRF</code>, <code>FNL</code> or <code>FPL</code>. Using <code>POS</code> or <code>TSP</code> is not allowed.</li> </ul>
<code>RON?</code>	<code>RON? [{&lt;AxisID&gt;}]</code>	Gets the referencing method.
<code>FRF</code>	<code>FRF [{&lt;AxisID&gt;}]</code>	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 is always done from the same side irrespective of the axis position when the command is sent.</li> <li>▪ 2: The approach is done via the negative limit switch.</li> </ul>
<code>FRF?</code>	<code>FRF? [{&lt;AxisID&gt;}]</code>	Queries whether referencing for an axis has already been done. 1 = Referencing has been done 0 = Referencing has not been done
<code>FNL</code>	<code>FNL [{&lt;AxisID&gt;}]</code>	Starts a reference move to the negative limit switch.
<code>FPL</code>	<code>FPL [{&lt;AxisID&gt;}]</code>	Starts a reference move to the positive limit switch.
<code>POS</code>	<code>POS {&lt;AxisID&gt; &lt;Position&gt;}</code>	Sets the current axis position (does not trigger motion) to reference the axis.
<code>POS?</code>	<code>POS? [{&lt;AxisID&gt;}]</code>	Gets the current axis position.

Com-mand	Syntax	Function
TSP	TSP {<InputSignalID> <Position>}	Sets the current sensor position (does not cause motion). If referencing is done in open-loop operation: TSP can be used to adjust the sensor position to the current axis position.
TSP?	TSP? [{<InputSignalID>}]	Gets the current sensor position.

### Parameters

Reference moves can be configured with the following parameters:

Parameters	Description and Possible Values
<b>Closed-Loop Deceleration (Phys. Unit/s<sup>2</sup>)</b> 0xC	Deceleration Used in both open-loop and close-loop operation. For details, see "Generating a Dynamics Profile" (p. 25).
<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 Used in both open-loop and close-loop operation. For details, see "Generating a Dynamics Profile" (p. 25).
<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 during referencing, the maximum of this value should be as large as the value of parameter 0x49. 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 the built-in limit switch and the 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.

Sequence of the reference move:

1. The axis moves to the negative limit switch.
2. The axis moves the distance away from the limit switch as specified by parameter 0x78 .
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. 34).

#### **INFORMATION**

For reference moves, you can also use the digital inputs of the C-663 as the source of the reference signal, the negative limit switch signal or the positive limit switch signal. See "Using Digital Input Signals as Switch Signals" (p. 99) for more information.

#### **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-663 are written to the nonvolatile memory in PIMikroMove or by entering the WPA command, the axis will no longer be considered "referenced" (the response to `FRF?` is 0).

## **4      Unpacking**

1. Unpack the C-663 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 (p. 273) immediately.
4. Keep all packaging materials in case the product needs to be returned.



# 5 Installation

## In this Chapter

Installing the PC Software .....	45
Mounting the C-663 .....	49
Grounding the C-663 .....	50
Connecting the Positioner .....	50
Connecting the PC .....	51
Connecting the Power Supply to the C-663 .....	54
Connecting an Analog Joystick .....	54
Connecting Digital Inputs and Outputs .....	55
Connecting Analog Signal Sources .....	57

### 5.1 Installing the PC Software

The communication between the C-663 and a PC is necessary to configure the C-663 and send motion commands using the commands of the GCS. Various PC software applications are available for this purpose.

#### 5.1.1 Doing Initial Installation

##### Accessories

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

##### Installing the PC software in Windows

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

The **InstallShield Wizard** window opens for installing the PC software.

2. Follow the instructions on the screen.

The PI software suite includes the following components:

- Driver for use with NI LabVIEW software
- Dynamic program library for GCS
- PIMikroMove

- PC software for updating the firmware of the C-663
- PI Update Finder for updating the PC software
- USB driver

### Installing the PC Software in Linux

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

You can select individual components for installation.

## 5.1.2 Installing Updates

PI is constantly improving the PC software.

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

### Requirements

- ✓ Active connection to the Internet.
- ✓ If your PC uses a Windows operating system:
  - You have downloaded the PI Update Finder manual (A000T0028) from the PI website. The link is in the "A000T0081-Downloading Manuals from PI.pdf" file in the \Manuals directory on the PI software CD.
- ✓ If your PC uses a Linux operating system:

You have the access data (user name and password) for the PI software CD. Information regarding the access data can be found in the file "C-990.CD1\_Releasenews.pdf" in the main directory or in \Manuals folder on the PI software CD.

### Updating the PC software and PISTAGES3.DB in Windows

- Use the PI Update Finder:
  - Follow the instructions in the PI Update Finder manual (A000T0028).

### Updating the PC software in Linux

1. Open the website <https://www.physikinstrumente.com/en/products/motion-control-software/> (<http://www.>).
2. Click **Login**.
3. Log in with the user name and password for the PI software CD.
4. Scroll down to **Downloads**.
5. Click the archive file "CD Mirror" or the associated download link.
6. Select the option in the following request to save the file to your PC.  
If you do not specify anything else, the "CD Mirror" archive file is stored in the default download directory of your PC.
7. Unpack the archive file into a separate installation directory.
8. Go to the **linux** subdirectory in the directory with the unpacked files.
9. Unpack the archive file in the **linux** directory by entering the command `tar -xvpf <name of the archive file>` on the console.
10. Read the accompanying information on the software update (readme file and/or "C-990.CD1\_Releasenews.pdf file) and decide whether the update makes sense for your application.
  - If no: Stop the update procedure.
  - If yes: Go through the following steps.
11. Log into the PC as superuser (root privileges).
12. Install the update.

#### **INFORMATION**

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

- Contact our customer service department (p. 273).

### Updating PISTAGES3.DB in Linux

1. Contact the customer service department (p. 273) to get the latest version of the PISTAGES3.DB positioner database.
2. Log into the PC as superuser (root privileges).
3. Install the update that you received from our customer service department onto your PC.

### 5.1.3 Installing Custom Positioner Databases

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

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

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

- Install the custom positioner database by double-clicking the file ***Import\_PI\_CustomStage.exe*** in the root directory of the CD.

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

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

- a) Update the PIStages3 database on your PC, see "Installing Updates" (p. 46).
- b) Repeat the installation of the custom positioner database.

## 5.2 Mounting the C-663

The C-663 can be used as benchtop device or mounted in any orientation on a surface.

The C-663 is stackable and can be installed in a control cabinet.

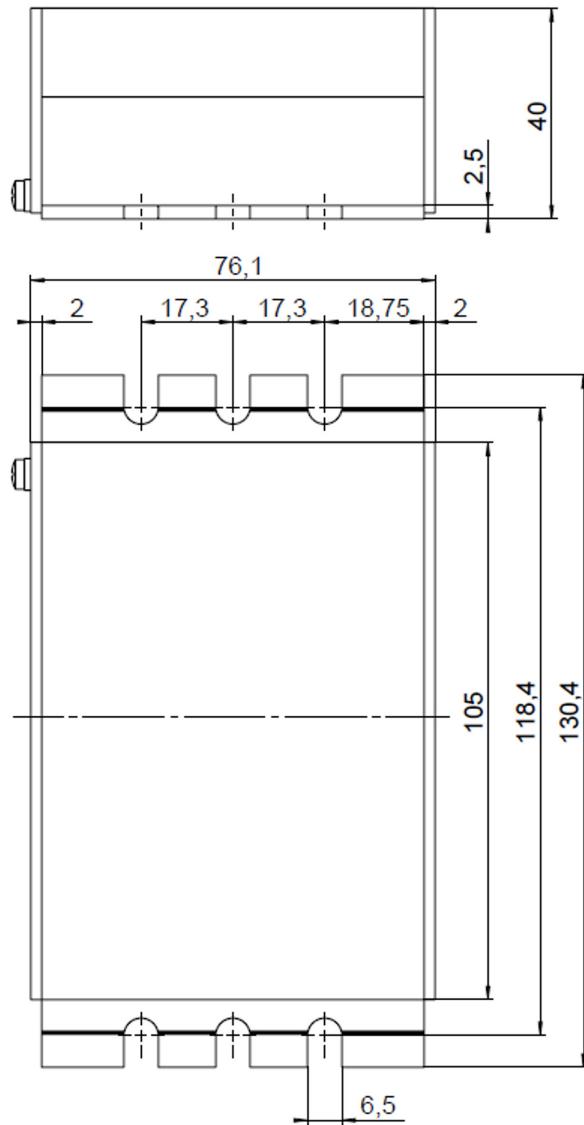


Figure 10: C-663, dimensions in mm

### Tools and accessories

- Suitable screws
- Suitable screwdriver

### Mounting the C-663 onto a surface

1. Make the necessary holes in the surface.  
The arrangement of the recesses in the mounting rails of the C-663 can be found in the figure above.
2. Use two screws on each side to affix the C-663 to the recesses in the mounting rails.

## 5.3 Grounding the C-663



The C-663 is not grounded via the power supply connection.

If a potential equalization is required:

- Connect the screw on the rear panel of the C-663's housing marked with the protective earth symbol (see figure) to the grounding system.

## 5.4 Connecting the Positioner

### INFORMATION

The C-663.12 supports positioners without sensor, and positioners equipped with an incremental position sensor. If the positioner does not have a sensor, an external incremental position sensor can be connected to the **Motor** socket (pins 19, 20, 21, 22).

### INFORMATION

When the positioner, cable, and C-663 are marked as a system:

- Contact our customer service department (p. 273) before exchanging system components.

### Requirement

- ✓ The C-663 is switched off, i.e., the power adapter is **not** connected to the power socket with the power cord.
- ✓ You have read and understood the user manual for the positioner.

### Tools and accessories

- Positioner with 2-phase stepper motor
- Suitable cable from the scope of delivery of the positioner

- If the positioner has a D-sub 15 connector: C-815.AA42 connector adapter for D-sub 15 (f) to HD D-sub 26 (m), included in the scope of delivery (p. 12)
- If the distance between C-663 and positioner is too long: Motor cable C-815.LSM2, 3 m, available as optional accessory (p. 12)
- If a position sensor is to be used that is **not** incorporated in the positioner: Suitable adapter that joins the lines of the positioner and the sensor, e.g., a Y cable.

### Connecting the Positioner

1. Connect the positioner to the **Motor** socket of the C-663.
2. Use the integrated screws to secure the connections against accidental disconnection.

## 5.5 Connecting the PC

Communication between the C-663 and a PC is required to configure the C-663 and to command motion using the GCS commands. The C-663 has the following interfaces for this purpose:

- RS-232 interface
- USB interface

In this section, you learn how to establish proper cable connections between the C-663 and a PC and in a daisy chain network. The steps for establishing communication between C-663 and PC are described in the section "Startup":

- "Establishing Communication via RS-232" (p. 63)
- "Establishing Communication via USB" (p. 64)
- "Establishing Communication for Networked Controllers" (p. 66)

#### **INFORMATION**

Using a daisy chain network, up to 16 controllers can be connected to the PC via a single RS-232 or USB connection.

## 5.5.1 Connecting to the RS-232 Interface

### NOTICE



#### Incorrect wiring!

Connecting the USB and RS-232 interfaces of the controller to the PC at the same time can damage the PC or the controller.

- Connect either the USB or the RS-232 interface to the PC.

#### Requirements

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

#### Tools and accessories

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

#### Connecting the C-663 to the PC

- Connect the **RS-232 In** socket on the front panel of the C-663 and the RS-232 interface of the PC (a D-sub 9(m) panel plug) to the null-modem cable.

## 5.5.2 Connecting to the USB Interface

### NOTICE



#### Incorrect wiring!

Connecting the USB and RS-232 interfaces of the controller to the PC at the same time can damage the PC or the controller.

- Connect either the USB or the RS-232 interface to the PC.

#### Requirements

- ✓ The PC has a free USB interface.

#### Tools and accessories

- USB cable (type A to mini-B) for connection to the PC, in the scope of delivery (p. 12)

#### Connecting the C-663 to the PC

- Connect the USB socket of the C-663 and the USB interface of the PC with the USB cable.

### 5.5.3 Building a Daisy Chain Network

#### **INFORMATION**

Networking in a daisy chain is done in series. See also "Definition of Terms" (p. 2). The first controller is connected directly to the PC.

#### **INFORMATION**

The DIP switches of the C-663 must be set accordingly:

- Set a unique address for each controller in a daisy chain network. One of the controllers must have the address 1. This controller does not have to be the one directly connected to the PC. See "Controller Address" (p. 61) for details.
- Set the same baud rate for every controller in a daisy chain network. See "Baud Rate" (p. 62) for details.

#### Tools and accessories

- A network cable for every controller to be connected to the network. Currently available:
  - C-862.CN, 30 cm, included in the scope of delivery
  - C-862.CN2, 180 cm, available as an optional accessory (p. 12)

#### Networking the controllers

- Set up the controller chain. For this purpose, always connect the **RS-232 Out** connection of the previous controller to the **RS-232 In** connection of the subsequent controller via the network cable.
- Connect the first controller of the chain to the PC.
  - Use the RS-232 interface (p. 52).
  - or**
  - Use the USB interface (p. 52).

#### **INFORMATION**

A C-663 can be operated in a common daisy chain network with the following controllers:

- Mercury DC Motor Controller C-863.11
- PILine® Motion Controller the series C-867
- PiezoWalk® NEXACT® Controller E-861

## 5.6 Connecting the Power Supply to the C-663

### **INFORMATION**

If the positioner is equipped with a PWM amplifier that is supplied via a separate power adapter:

- To achieve the optimum motor performance, use a power adapter for the C-663 that supplies the same output voltage as the power adapter for the PWM amplifier.

### **Requirements**

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

### **Tools and accessories**

- Wide input range power supply (p. 12) included, for line voltages between 100 and 240 volts alternating voltage at 50 or 60 Hz
- Alternative: Sufficiently rated power adapter
- Power cord supplied
- Alternative: Sufficiently sized power cord

### **Connecting the power adapter to the C-663**

- Connect the 4-pin connector of the power adapter to the **48 V 2 A** socket of the C-663.
  - Make sure that the connector is locked in the socket.
- Connect the power cord to the power adapter.

## 5.7 Connecting an Analog Joystick

### **INFORMATION**

You can connect an axis and a button of an analog joystick to the **Joystick** socket:

- Pin 4: Axis 1 of joystick 1
- Pin 6: Button 1 of joystick 1

You can use joystick axis to control the velocity of the positioner connected to the C-663.

### **INFORMATION**

The C-819.20 and C-819.30 joysticks, available as optional accessories, use pins 4 and 6 of the **Joystick** socket. Pin 3 of this socket is used as power supply of the joystick.

You can use a C-819.20Y Y cable to connect two C-663 to a C-819.20 joystick. In this case, power is supplied to the joystick by the C-663, which is connected to the X branch of the cable.

### Tools and accessories

- Analog joystick from PI for operation with 0 to 3.3 V, available as an optional accessory (p. 12):
  - C-819.20 analog joystick for 2 axes
  - If a C-819.20 joystick is to be connected to two controllers: C-819.20Y Y cable or
  - C-819.30 analog joystick for 3 axes

### Connecting an analog joystick

- Connect the joystick to the **Joystick** socket of the C-663:
  - If you want to operate a C-819.20 joystick with this controller only, connect it directly to the controller.
  - If you want to operate a C-819.20 joystick with two controllers (i.e., two axes), connect the joystick to the C-819.20Y Y cable and connect both controllers to the X and Y branches of the cable. The power is supplied to the joystick via the X branch. For this reason, the X branch has to be connected to a controller even if joystick control is not to be activated for this controller.
  - If you want to connect an axis of a C-819.30 joystick, connect the corresponding cable of the joystick to the controller.

## 5.8 Connecting Digital Inputs and Outputs

The digital inputs and outputs on the **I/O** socket of the C-663 can be used as follows:

- Outputs: Triggering of external devices; see "Digital Output Signals" (p. 89).
- Inputs: Use in macros (p. 99) and/or as source for the reference and limit switch signals of the axis (p. 99)

### 5.8.1 Connecting the Digital Outputs

#### **INFORMATION**

Digital output signals are available on pins 5, 6, 7 and 8 of the **I/O** socket.

#### **INFORMATION**

If the C-170.PB pushbutton box from PI is connected to the **I/O** socket, it displays via LEDs the state of the digital output lines.

### Tools and accessories

- Suitable cable, e. g. C-170.IO IO cable with open end, available as an optional accessory (p. 12)
- Device to be triggered having digital input for TTL signals

### Connecting a device to be triggered

- Connect an appropriate device to one of pins 5, 6, 7, and 8 of the **I/O** socket of the C-663.

## 5.8.2 Connecting the Digital Inputs

### **INFORMATION**

Digital input signals can be fed via pins 1, 2, 3, and 4 of the **I/O** socket into the C-663.

### **INFORMATION**

The digital inputs (pins 1 to 4) on the **I/O** socket can also be used as analog inputs.

- Digital: TTL
- Analog: 0 to +5 V

### Tools and accessories

- Suitable signal source:
  - If the digital inputs are to be used in macros, the C-170.PB pushbutton box, for example, can be connected, available as an optional accessory (p. 12).
  - If the digital inputs are to be used as the source for the reference and limit switch signals of the axis, the signal level may only change once across the entire travel range.
- If necessary: Suitable cable, e. g. C-170.IO IO cable with open end, available as an optional accessory (p. 12).

### Connecting a digital signal source

- Connect an appropriate signal source to one of pins 1, 2, 3, or 4 of the **I/O** socket of the C-663.

## 5.9 Connecting Analog Signal Sources

The analog inputs on the **I/O** socket of the C-663 can be used as follows:

- Use in macros (p. 102): Details and examples of macros are found in "Controller Macros" (p. 110).
- Scanning applications with PIMikroMove (see PIMikroMove manual)

### **INFORMATION**

Analog input signals can be fed via pins 1, 2, 3, and 4 of the **I/O** socket into the C-663.

### **INFORMATION**

The analog inputs (pins 1 to 4) on the **I/O** socket can also be used as digital inputs.

- Analog: 0 to +5 V
- Digital: TTL

### Tools and accessories

- Suitable signal source
- If necessary: Suitable cable, e. g. C-170.IO IO cable with open end, available as an optional accessory (p. 12).

### Connecting an analog signal source

- Connect an appropriate signal source to one of pins 1, 2, 3 or 4 of the **I/O** socket of the C-663.



# 6 Startup

## In this Chapter

General Notes on Startup .....	59
Adapting the DIP Switch Settings.....	60
Switching the C-663 On .....	62
Establishing Communication.....	63
Starting Motion.....	71
Optimizing the Servo Control Parameters.....	76
Configuration Steps for Advanced Users .....	82

### 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.

- Do **not** allow the C-663 to disable evaluation of the limit switch signals via the parameter setting.
- Check the function of the limit switches at about 10 % to 20 % of the maximum velocity.
- Stop the motion immediately if the limit switches malfunction.

#### INFORMATION

The factors for the conversion of microsteps and counts into the desired physical unit of length determine the values of all other length-based parameters of the C-663. When adapting the C-663's parameters to the connected positioner, parameters 0xE, 0xF, 0x9A, and 0x9B must therefore be set before all other parameters.

If you load the parameters from a positioner database (e.g., when working with PIMikroMove), the correct sequence is maintained automatically.

## 6.2 Adapting the DIP Switch Settings

### 6.2.1 General Procedure

#### **INFORMATION**

Changed DIP switch settings become effective after the C-663 is switched on.

- If you have changed the DIP switch settings while the C-663 was switched on, switch the C-663 off and back on again to activate the new settings.

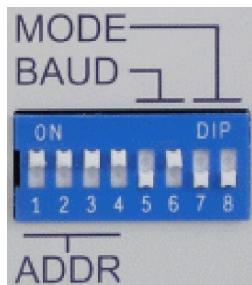


Figure 11: DIP switches: switch up = ON; switch down = OFF

Switch	Function
1 to 4	Controller address (p. 61)
5 and 6	Baud rate (p. 62)
7 and 8	Without function

#### Requirements

- ✓ The C-663 is switched off, i.e., the power supply is **not** connected to the power socket with the power cord.

#### Adapting the DIP switch settings

- Put the individual DIP switches in the correct position for your application. Details are specified in the following tables.

## 6.2.2 Controller Address

Address*	S1	S2	S3	S4
<b>1</b>	<b>ON</b>	<b>ON</b>	<b>ON</b>	<b>ON</b>
2	ON	ON	ON	OFF
3	ON	ON	OFF	ON
4	ON	ON	OFF	OFF
5	ON	OFF	ON	ON
6	ON	OFF	ON	OFF
7	ON	OFF	OFF	ON
8	ON	OFF	OFF	OFF
9	OFF	ON	ON	ON
10	OFF	ON	ON	OFF
11	OFF	ON	OFF	ON
12	OFF	ON	OFF	OFF
13	OFF	OFF	ON	ON
14	OFF	OFF	ON	OFF
15	OFF	OFF	OFF	ON
16	OFF	OFF	OFF	OFF

\*Factory default settings are shown in bold.

### INFORMATION

A unique address must be set for each controller in a daisy chain network. One of the controllers must have the address 1. This controller does not have to be the one directly connected to the PC.

### INFORMATION

A non-networked controller must have address 1, if it

- is to be used in PI MikroMove.
- is to be used in drivers for NI LabVIEW software.
- is to be addressed with PI Terminal without specifying the target address; in this case, the target and sender address (p. 132) are also not specified in the responses from the C-663.

## 6.2.3 Baud Rate

Baud rate*	S5	S6
9600	ON	ON
19200	ON	OFF
38400	OFF	ON
<b>115200</b>	<b>OFF</b>	<b>OFF</b>

\*Factory default settings are shown in bold.

### INFORMATION

The same baud rate must be set for all controllers in a daisy chain network.

## 6.3 Switching the C-663 On

### INFORMATION

After switching on the C-663 the motor is deactivated by default.

- Query the activation state of the motor with the `SVO?`, `#4` or `SRG?` commands.
- Activate the motor with the `SVO` command.
- If necessary, program a startup macro that starts the C-663 via the `SVO` command with activated motor; refer to "Configuring a Startup Macro" (p. 119).

### Requirements

- ✓ You have read and understood the general notes on startup (p. 59).
- ✓ The C-663 has been installed properly (p. 45).
- ✓ You have set the DIP switches on the C-663 in accordance with your application (p. 60).

### Switching the C-663 on

- Plug the power cord of the power adapter into the power socket.

The C-663 loads the parameter values from the nonvolatile memory into the volatile memory.

The **STA** LED on the front panel of the C-663 displays the state of the C-663:

- Lights up continuously: C-663 is ready for normal operation
- Flashing: C-663 is in firmware update mode

- Off: C-663 is not connected to the power supply or could be defective
- If the C-663 is in firmware update mode, update the firmware (p. 263).
- If the C-663 is properly connected to the power adapter (p. 54) and the **STA** LED does not light up after switching on, contact our customer service department (p. 273).

## 6.4 Establishing Communication

The procedure for PIMikroMove is described in the following.

### INFORMATION

Use the **USB Daisy Chain** and **RS-232 Daisy Chain** tabs in the PC software for establishing communication only if you have actually connected a daisy chain network to the PC.

### INFORMATION

A non-networked controller must have the address 1, if it is to be used in PIMikroMove. See "Controller Address" (p. 61) for details.

### 6.4.1 Establishing Communication via RS-232

#### Requirements

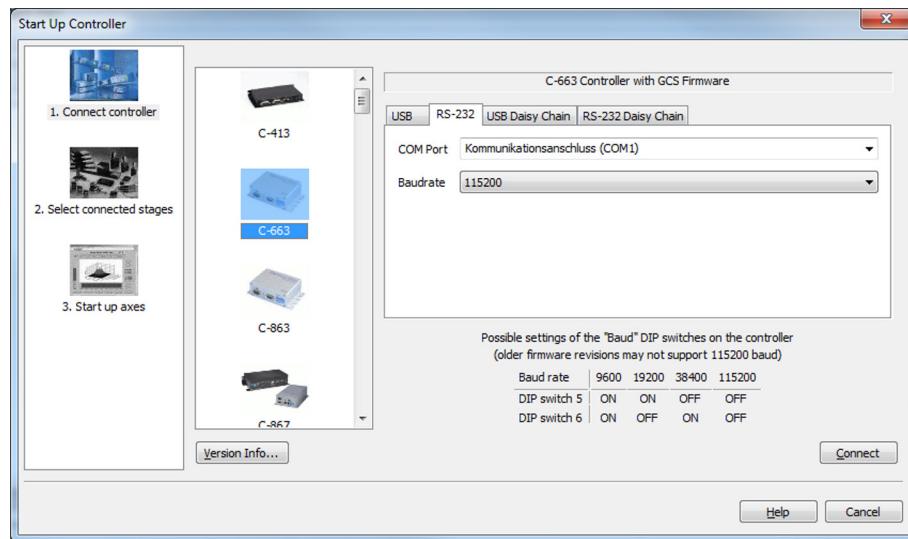
- ✓ You have read and understood the general notes on startup (p. 59).
- ✓ The C-663 is connected to the RS-232 interface of the PC (p. 52).
- ✓ You have made the following settings with the respective DIP switches prior to switching on the C-663 (p. 60, p. 60):
  - controller address = 1
  - appropriate baud rate
- ✓ The C-663 is switched on (p. 62).
- ✓ The PC is switched on.
- ✓ The required software is installed on the PC (p. 45).
- ✓ You have read and understood the manual for the PC software. The links to the software manuals are in the A000T0081 file on the PI software data storage medium.

#### Establishing communication

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-663** in the controller selection field.
3. Select the **RS-232** tab on the right-hand side of the window.
4. In the **COM Port** field, select the COM port of the PC to which you have connected the C-663.
5. In the **Baudrate** field, set the value which is set with DIP switches 5 and 6 of the C-663.  
This adapts the baud rate of the PC to the baud rate of the C-663.
6. Click **Connect** to establish communication.

If communication was established successfully, PIMikroMove guides you through the configuration of the C-663 for the connected positioner; see "Starting Motion" (p. 71).

#### 6.4.2 Establishing Communication via USB

##### 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. 59).
- ✓ The C-663 is connected to the USB interface of the PC (p. 52).

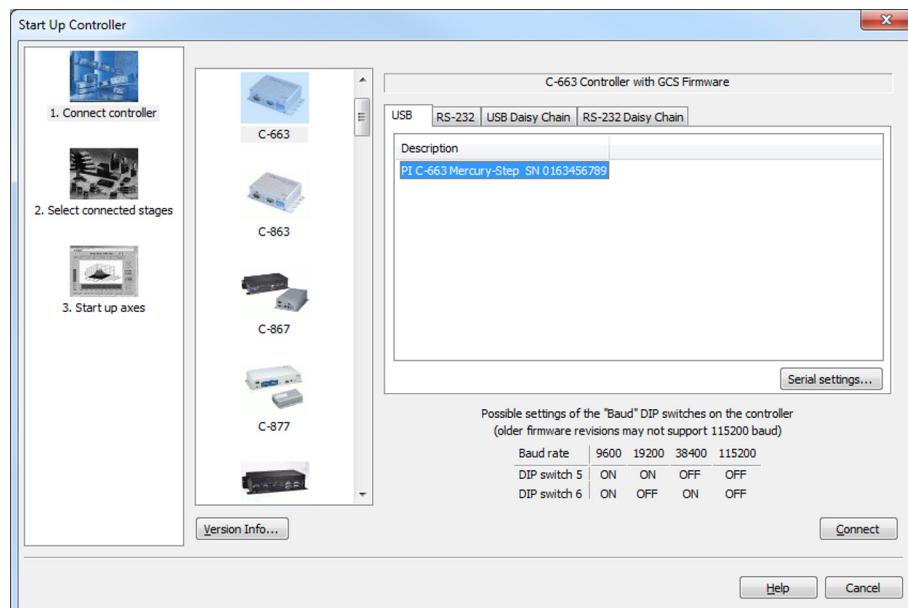
- ✓ Prior to switching on the C-663 you set the DIP switches for the controller address to address 1 (p. 60, p. 60).
- ✓ The C-663 is switched on (p. 62).
- ✓ The PC is switched on.
- ✓ The required software and USB drivers are installed on the PC.
- ✓ You have read and understood the manual for the PC software. The links to the software manuals are in the A000T0081 file on the PI software data storage medium.

### Establishing communication

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-663** in the controller selection field.
3. Select the **USB** tab on the right-hand side of the window.
4. Select the connected C-663 in the **USB** tab.
5. Click **Connect** to establish communication.

If communication was established successfully, PIMikroMove guides you through the configuration of the C-663 for the connected positioner; see "Starting Motion" (p. 71).

- If communication could not be established, look for a solution to the problem in "Troubleshooting" (p. 269).

### 6.4.3 Establishing Communication for Networked Controllers

The following describes the procedure for PIMikroMove and for PITerminal.

#### INFORMATION

If you are establishing communication with a networked controller via PITerminal, the address of the controller to be addressed is required in every command line. See "Target and Sender Address" (p. 132) for details.

- Use PITerminal to test communication with networked controllers.

#### INFORMATION

The RS-232 output lines of some PCs are not adapted for the maximum number of 16 controllers in a network. If you have connected a daisy chain network to such a PC via the RS-232 interface, communication malfunctions may occur (e.g., timeout). In case of communication malfunctions:

1. Disconnect the null-modem cable from the **RS-232 In** socket of the controller which is connected to the PC.
2. Connect the daisy chain network to the PC via the USB interface of this controller.

#### Requirements

- ✓ You have read and understood the general notes on startup (p. 59).
- ✓ You have set up a daisy chain network .
- ✓ You have assigned (p. 61)a unique address to each of the networked controllers.
- ✓ You have set (p. 62) the same baud rate for all controllers.
- ✓ All controllers in the daisy chain network are switched on (p. 62).
- ✓ The PC is switched on.
- ✓ The required software is installed on the PC (p. 45).
- ✓ If you have connected the first controller in the chain to the PC via the USB interface: The USB drivers are installed on the PC (p. 45).
- ✓ You have read and understood the manual for the PC software. The links to the software manuals are in the A000T0081 file on the PI software data storage medium.

#### Establishing communication with PIMikroMove

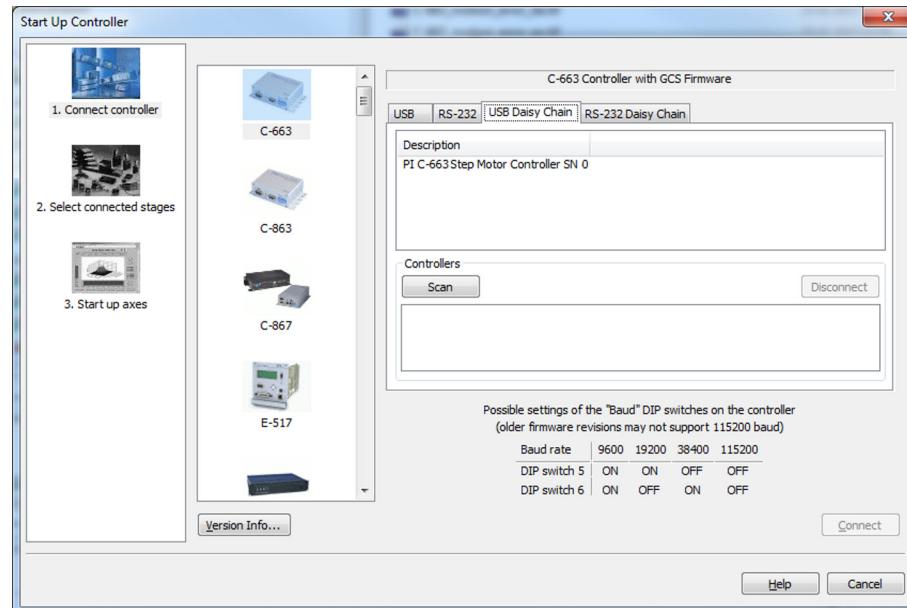
1. Start PIMikroMove.

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

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

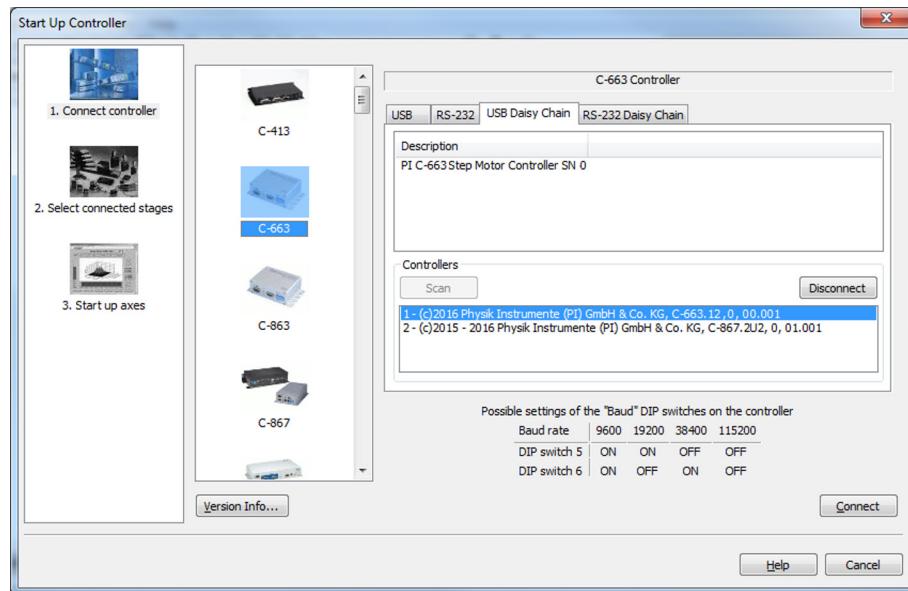
2. Select the appropriate controller type in the field for controller selection.

In the example in the following figures, there is a daisy chain network from a C-663.12 with the controller address 1 and a C-867.2U2 with the controller address 2. If you want to connect the C-663.12 first, select **C-663**.



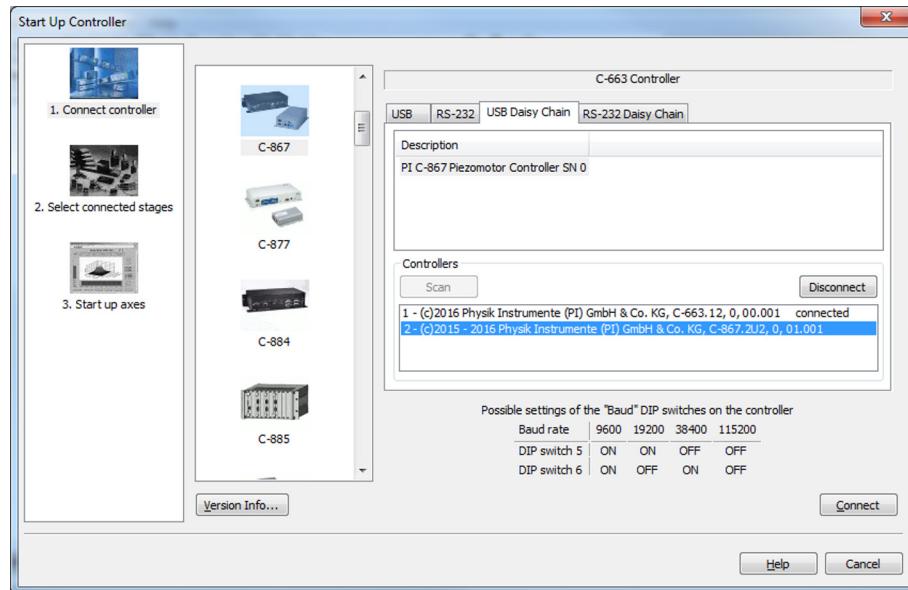
3. Select the appropriate tab on the right-hand side of the window:
  - If you have connected the first controller in the chain to the PC via the RS-232 interface, select the **RS-232 Daisy Chain** tab.
  - If you have connected the first controller in the chain to the PC via the USB interfaces, select the **USB Daisy Chain** tab.
4. Make the settings for the interface in the tab selected:
  - **RS-232 Daisy Chain** tab:
    - Select the COM port of the PC in the **COM Port** field which you connected to the first controller in the chain.
    - Set the value in the **Baudrate** field that is set for all controllers in the chain.
  - **USB Daisy Chain** tab:
    - Select the controller in the upper part of the tab connected to the PC.

- Click the **Scan** button in the bottom section of the tab to list every controller in the daisy chain network.



- Select a controller from the list. The selection must match the controller type that you selected in step 2.
  - Click **Connect** to establish communication with the controller selected.
- When communication has been successfully established, PIMikroMove guides you through the configuration of the C-663.12 for the connected positioner.
- Proceed further as described in "Starting Motion" (p. 71).
- If you want to connect an additional controller of the daisy chain network, select the **Connections > New...** menu item in the main window.
  - Do steps 2, 6, and 7 once again in the specified order.

In the following figure, the **C-867** is to be connected as well.



10. Repeat steps 8, 2, 6 and 7 for every additional controller of the daisy chain network, which you want to connect.

If you want to terminate communication with one of the controllers of the daisy chain network:

- Select the **Connections > Close** menu item for the corresponding controller in the main window.

### Establishing communication with PITerminal

#### **INFORMATION**

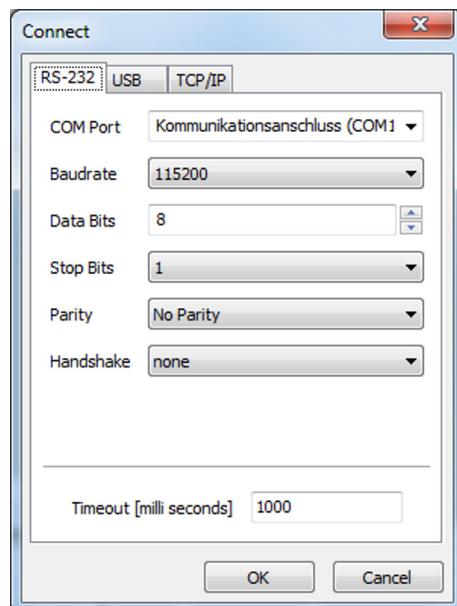
Via the **Mercury** button PITerminal supports controllers with older firmware versions that are not compatible with GCS.

- Make sure that the **Mercury** button is **not** activated in PITerminal.

1. Start PITerminal.
2. Click on **Connect....**

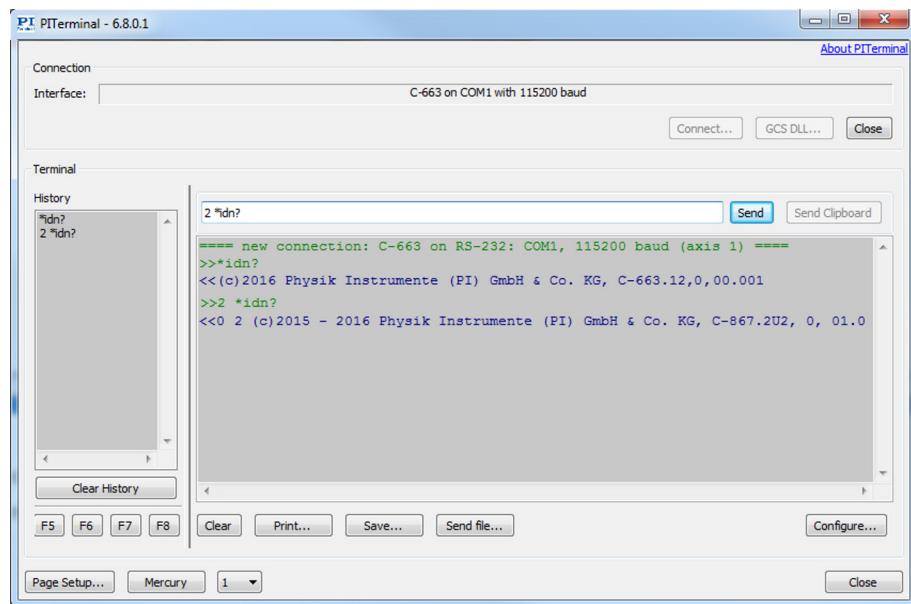
The **Connect** window opens.

3. Select the **RS-232** or **USB** tab in the **Connect** window according to the interface you used to connect the first controller in the chain to the PC.



4. Make the settings for the interface in the selected tab:
  - **RS-232** tab:
    - Select the COM port of the PC connected to the C-663 in the **COM Port** field.
    - Set the value of the C-663 in the **Baudrate** field with DIP switches 5 and 6.
  - **USB** tab:
    - Select the C-663 connected.
5. Click **OK** to establish communication.
6. Send the **\*IDN?** command for every controller in the daisy chain network to check the communication.  
In the example in the following figure, the daisy chain network comprises a C-663.12 with the controller address 1 and a C-867.2U2 with the controller address 2. Send:
  - **\*IDN?** to query the device identification string of the controller with the address 1; the controller address is not required (because = 1)
  - **2 \*IDN?** to query the device identification string of the controller with the address 2.

For further information, see "Target and Sender Address" (p. 132).



## 6.5 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-663 for the connected positioner
- Activating the motor
- Doing a reference move; details see "Referencing" (p. 38)

### 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-663'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 positioner is oscillating (unusual operating noise), immediately turn off the motor or disconnect the C-663 from the power supply.
- Activate the motor again after you have modified the servo control parameter settings of the C-663; see "Optimizing Servo Control Parameters" (p. 76).
- If, due to a very high load, oscillation occurs during the reference move, follow the instructions for the reference move in "Troubleshooting" (p. 269).

**INFORMATION**

The C-663 has a nonvolatile memory for parameter values. Therefore, after switching on, the correct parameter settings for the connected positioner may already be loaded.

- However, if you have loaded a parameter set from the positioner database and overwritten the original settings of the C-663 in the volatile memory during the process, avoid saving the new settings in the nonvolatile memory of the C-663. The original settings are active again after the C-663 has been switched off and on again or been rebooted.
- If you are using the ***Start up controller*** window in PIMikroMove for selecting the positioner and you are asked how you want to save the new settings, click the ***Save all settings permanently on controller*** button only if you are sure that the C-663 is working correctly with the settings.

**INFORMATION**

If the ***Select connected stages*** step is not displayed in PIMikroMove, the controller has probably already loaded the correct parameter settings for the positioner type connected.

1. Check ***Start up axes*** step to ensure that the correct positioner type is in the ***Stage*** column in the middle of the window.
2. If the positioner type is not correct, click ***Select connected stages*** in the left area of the ***Start up controller*** window to change the positioner type.

**INFORMATION**

The operating mode of the C-663 is set via the parameter ***Stepper Closed-Loop Operation*** (0x3101), for details see "Operating Modes" (p. 20). The parameter value for PI positioners is loaded from the positioner database in order to work in the appropriate operating mode:

- Positioners with sensor: Parameter value 1 = closed-loop operation
- Positioners without sensor: Parameter value 0 = open-loop operation

If you want to set the operating mode manually:

- Follow the instructions in "Setting the Operating Mode" (p. 82).

## Requirements

- ✓ You have read and understood the general notes on startup (p. 59).
- ✓ PIMikroMove is installed on the PC (p. 45).
- ✓ You have read and understood the PIMikroMove manual. The links to the software manuals are in the A000T0081 file on the PI software data storage medium.
- ✓ You have installed the latest version of the PISTAGES3.DB database onto your PC (p. 45).
- ✓ If PI provided a custom positioner database for your positioner, the dataset was imported into PIStages3 (p. 48).
- ✓ 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-663 to the positioner (p. 51).
- ✓ You have established communication with PIMikroMove between the C-663 and the PC (p. 63).

## Starting motion with PIMikroMove

1. When the **Select connected stages** step is displayed in PIMikroMove, select the type for the connected positioner:

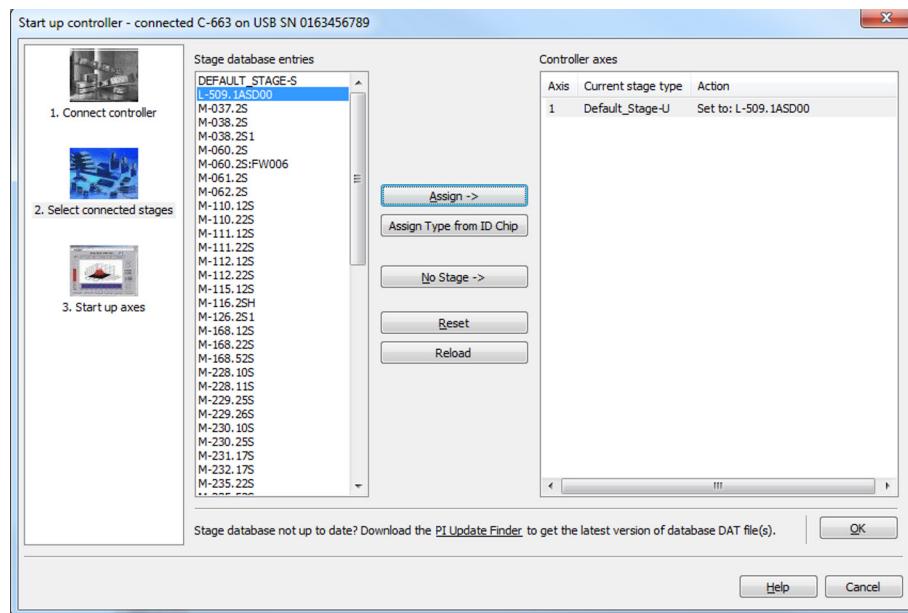
If the correct positioner type is already listed in the **Current stage type** column in the **Controller axes** list on the right of the window:

- Click **OK**.

If the listed positioner type is not correct:

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

- c) Confirm the selection with **OK**.



2. Specify how you want to load the parameter settings into the C-663 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-663. The settings are lost when the C-663 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-663. The settings are available immediately after switching on or rebooting the C-663 and do not need to be reloaded.

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

3. Do the reference move for the axis during the **Start up axes** step so that the controller knows the absolute axis position.

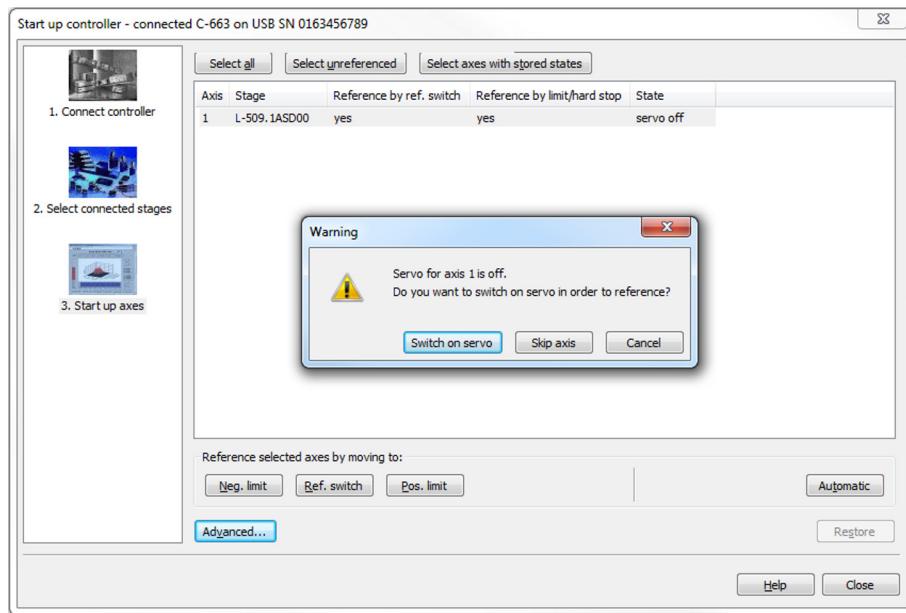
Options for starting the reference move:

- 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**.

Depending on the configuration of PIMikroMove, separate dialogs for start and result display of the reference move open.

If a warning message indicates that servo mode is switched off: In the case of the C-663.12, this means that the motor is deactivated.

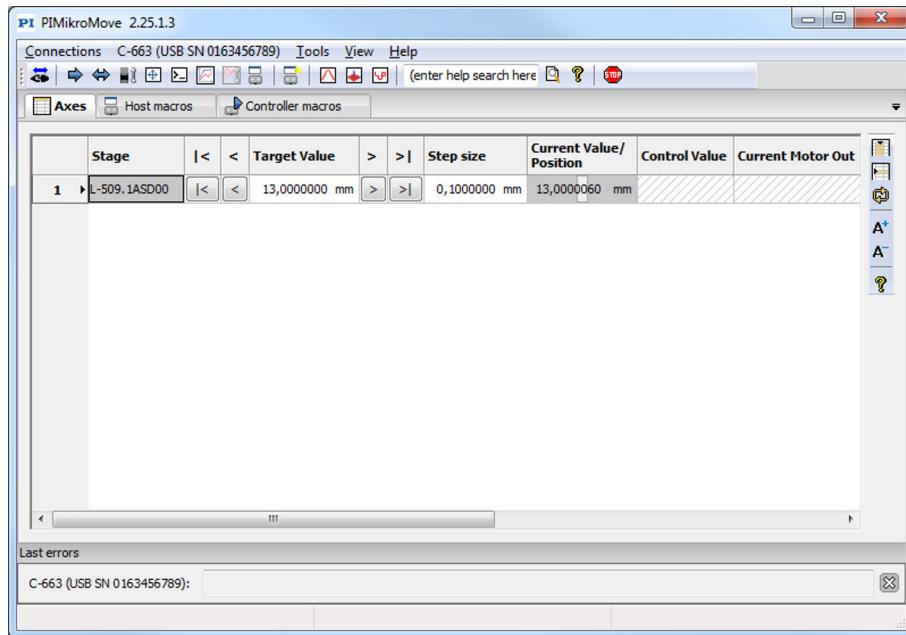
- Activate the motor by clicking the ***Switch on servo*** button.



The axis does the reference move.

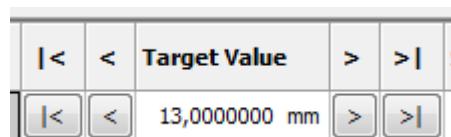
4. After a successful reference move, close the ***Start up controller*** window by clicking ***Close***.

The main window of PIMikroMove opens.



5. 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.



## 6.6 Optimizing the Servo Control Parameters

Adjusting the P-I-D controller during closed-loop operation improves the dynamic properties of the system (overshoot and settling time). The optimum P-I-D controller settings depend on your application and your requirements.

Typically, optimization is determined empirically, i.e., the behavior of the positioner is monitored with different settings in closed-loop operation. Optimization is done with the following parameters (for details see "Position Control" (p. 28)):

- **P Term 1 (Motion)** (0x3111)
- **I Term 1 (Motion)** (0x3112)
- **D Term 1 (Motion)** (0x3113)
- **I-Limit 1 (Motion)** (0x3114)
- **P Term 2 (End Position)** (0x3121)
- **I Term 2 (End Position)** (0x3122)
- **D Term 2 (End Position)** (0x3123)
- **I-Limit 2 (End Position)** (0x3124)

The following describes the procedure for optimizing the servo control parameters in PIMikroMove.

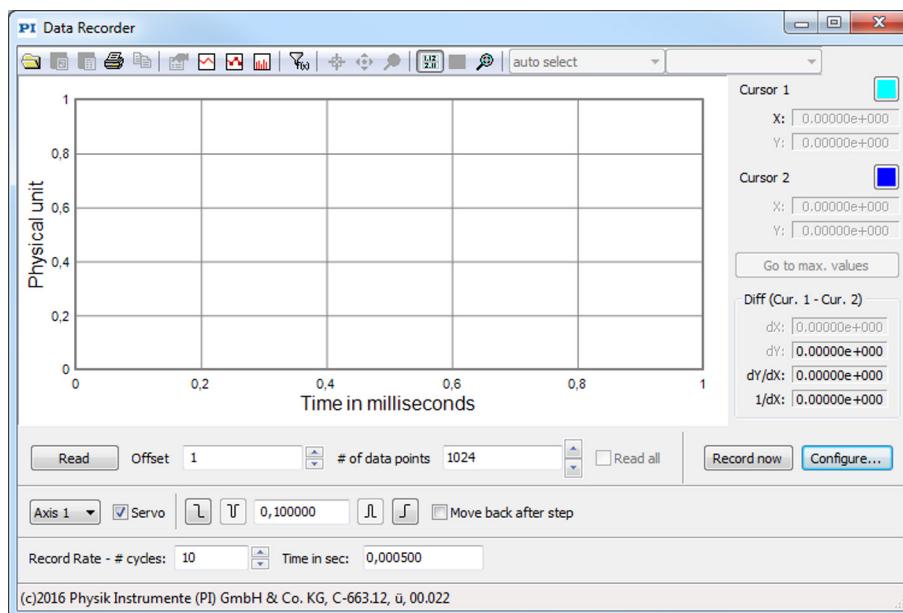
### Requirement

- ✓ The positioner has a sensor.
- ✓ You have installed the positioner in the same way as it will be used in your application (corresponding load, orientation, and fixing).
- ✓ The C-663 is set for closed-loop operation.  
Closed-loop operation is automatically set for PI positioners after selecting the positioner type, see "Starting Motion" (p. 71). For details on setting manually, see "Setting the Operating Mode" (p. 82).
- ✓ You started (p. 71) initial motion with PIMikroMove.
- ✓ All devices are still ready for operation.

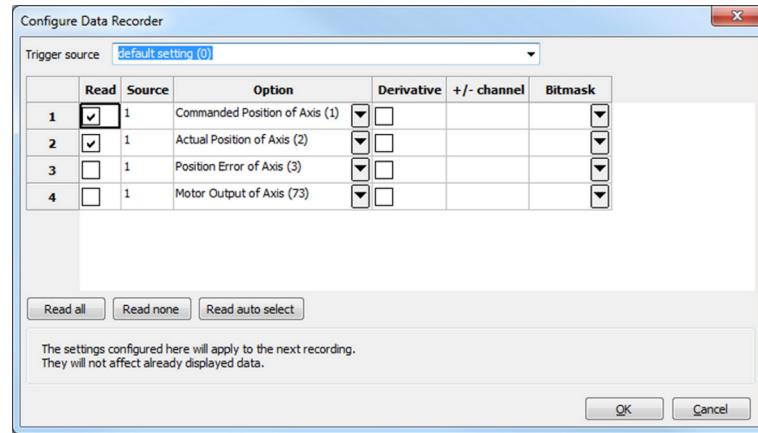
### Checking the servo control parameters: Record the step response

With the recording of the step response, you determine the settling behavior of the positioner in closed-loop operation.

1. Open the **Data Recorder** window in the main window of PIMikroMove via the **C-663 > Show data recorder** menu item.
2. Use the **Servo** checkbox to activate the motor for the axis (checkmark).
3. Configure the data recorder.
  - a) Set the size of the step to be made to a value that is typical for your application, e.g., 0.100000 (specified in physical units).
  - 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 field **# of data points**.



- 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**.



4. Start the jump in the positive direction as well as the recording by clicking the button in the **Data Recorder** window.
- The axis does the step and the step response is recorded and displayed graphically.
5. Check the displayed step response (see examples 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:

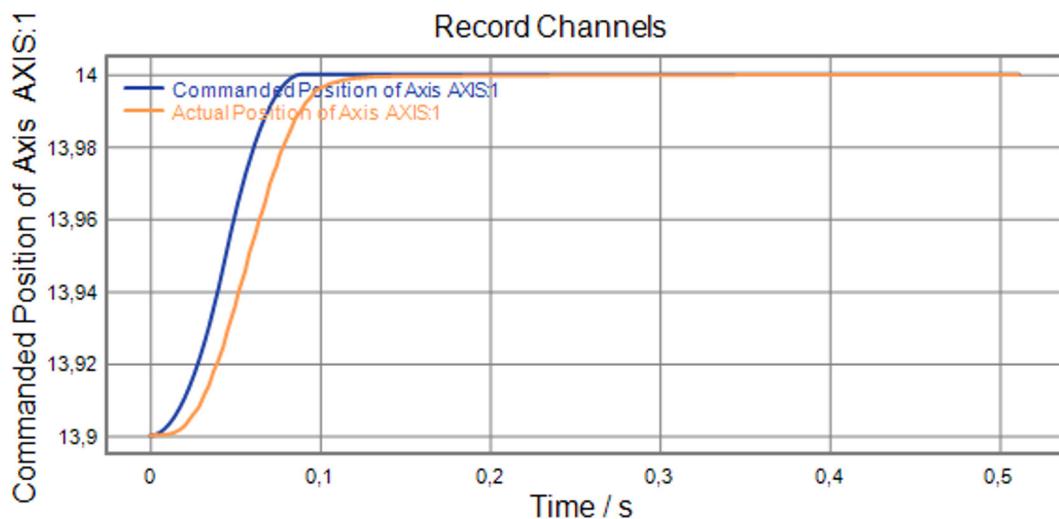


Figure 12: Step-and-settle too slow

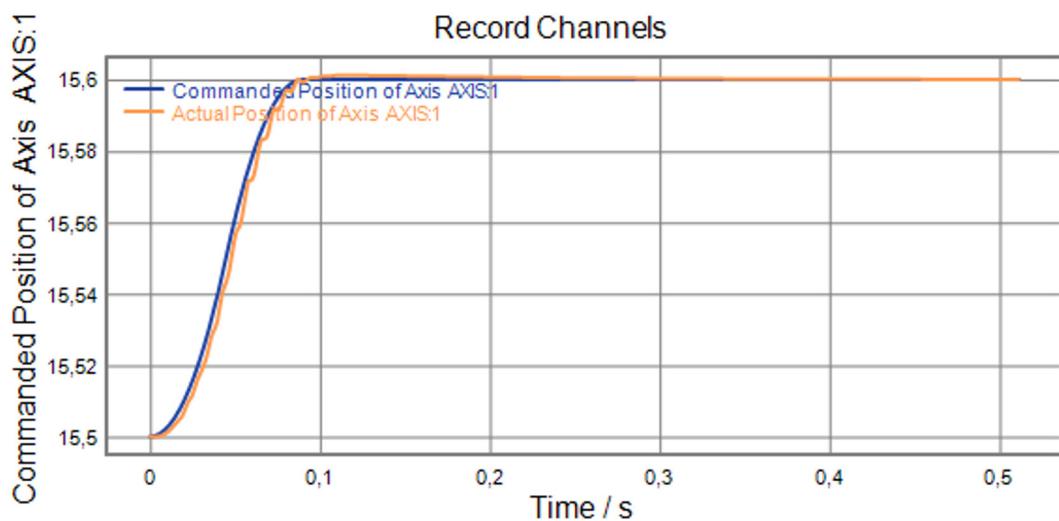


Figure 13: Oscillation

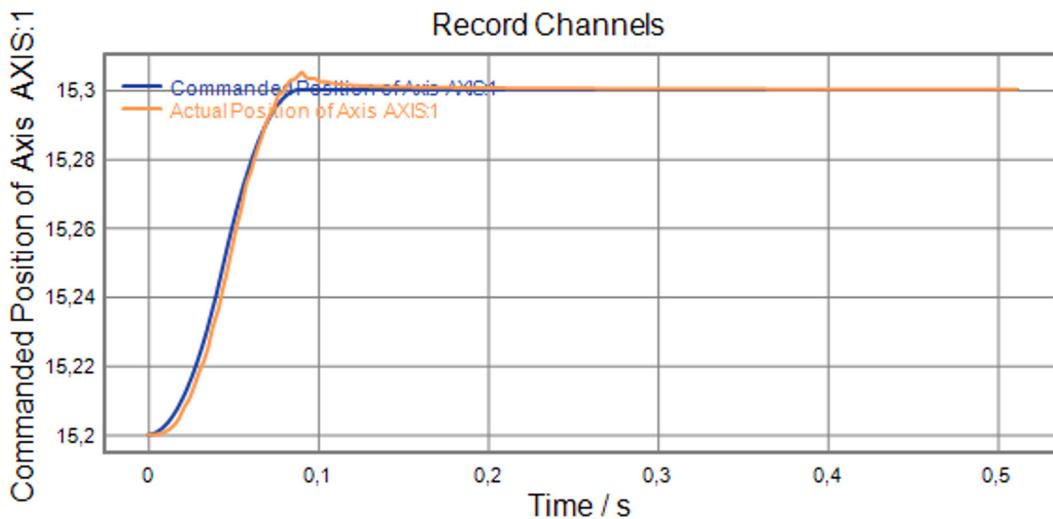


Figure 14: Strong overshooting

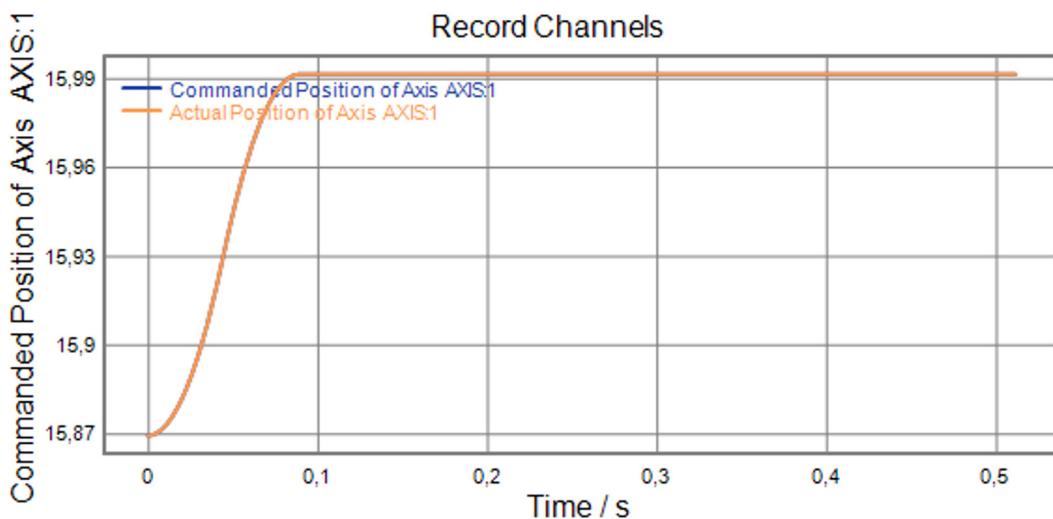


Figure 15: Optimal settling behavior (commanded and actual position congruent)

If the result is satisfactory (i.e., minimum overshoot, settling time not too long):

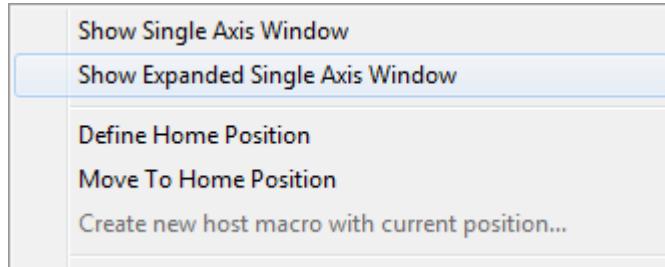
- You already have optimum parameter settings and do not have to do anything further.

If the result is not satisfactory:

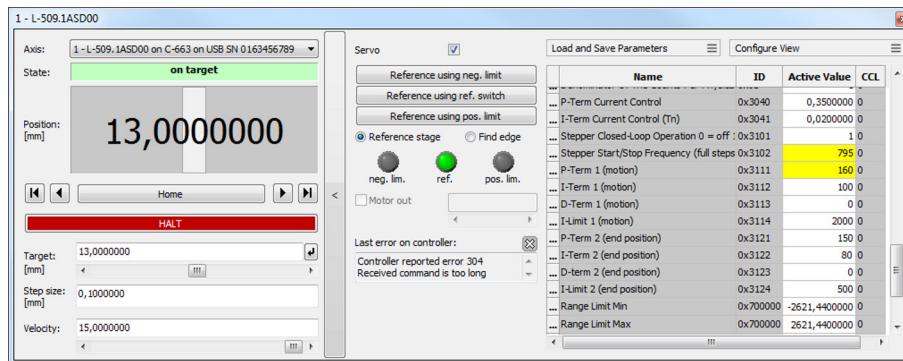
- Optimize the servo control parameters, see below.

### 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:
  - a) If the parameters to be changed are not included in the list on the right-hand of the window, click **Configure View -> Select parameters...** and add them to the list.
  - b) Type the new parameter value into the corresponding input field in the **Active Value** column of the list.
  - c) 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. Rerecord the positioner's step response in the **Data Recorder** window.

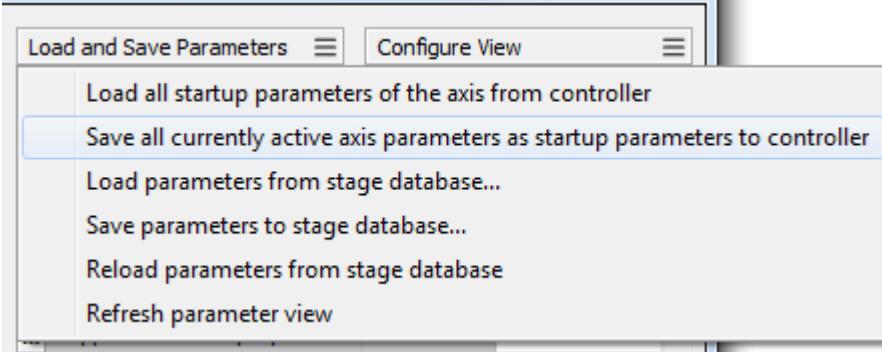
If the result is not satisfactory:

- 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 **Load and Save Parameters -> Save parameters to stage database...**, see "Creating or Modifying a Positioner Type" (p. 251).
- Transfer the current values of the listed parameters from the volatile memory to the nonvolatile memory of the C-663 by clicking **Load and Save Parameters -> Save**.

*all currently active axis parameters as startup parameters to controller.*



## 6.7 Configuration Steps for Advanced Users

### 6.7.1 Setting the Operating Mode

The C-663 supports the following operating modes, see "Operating Modes" (p. 20) for details:

- Operation with position control: "Closed-loop operation"
- Operation without position control: "Open-loop operation"

The operating mode is set with the parameter **Stepper Closed-Loop Operation** (0x3101).

#### INFORMATION

The value of the **Stepper Closed-Loop Operation** parameter for PI positioners is loaded from the positioner database, see "Starting Motion" (p. 71). An adjustment of the loaded parameter value is **not** necessary.

#### Requirements

- ✓ You have established (p. 63) communication between the C-663 and the PC with PIMikroMove or PITerminal.

#### Setting the operating mode

1. If you use PIMikroMove, open the window for sending commands:
  - a) When the **Start up controller** window is open, close it by clicking **Close** or **Cancel** to go to the main window.
  - b) Select the **Tools > Command entry** menu item in the main window or press **F4** on the keyboard.

In PITerminal the main window from which commands can be sent is opened automatically after establishing communication.
2. Query the current operating mode by sending `SPA? Send <AxisID> 0x3101.`

Possible answers:

- Open-loop operation: <AxisID> 0x3101=0
  - Closed-loop operation: <AxisID> 0x3101=1
- <AxisID> refers to the axis identifier.

3. If you want to change the operating mode:

- a) Make sure that the motor is deactivated by sending SVO <AxisID> 0.
- b) Set the desired operating mode:
  - If you want to change the current operating mode, send SPA <AxisID> 0x3101 1 for closed-loop operation or SPA <AxisID> 0x3101 0 for open-loop operation. The settings are lost when C-663 is switched off or rebooted.
  - If you want to change the default settings of the operating mode, send SEP 100 <AxisID> 0x3101 1 for closed-loop operation or SEP 100 <AxisID> 0x3101 0 for open-loop operation. The settings are activated with a reboot of the C-663 or by sending RPA <AxisID> 0x3101.

### 6.7.2 Open-loop Operation with Sensor: Position Adjustment

In some applications, it may be useful to operate the axis in the open-loop mode and to observe the position with a sensor. For details on the open-loop operation, see "Operating Modes" (p. 20).

In open-loop operation, the position measured by the sensor is **not** set to a defined value by a reference move. If the current position of the axis and the position value measured by the sensor are to be comparable, it is necessary to adjust the two position values after a reference move in open-loop operation.

#### **INFORMATION**

For PI positioners equipped with a sensor, the reference move is done in **closed-loop** operation. Adjustment of the position is **not** necessary after a reference move in closed-loop operation.

#### Requirements

- ✓ A position sensor is connected (p. 50) via the **Motor** socket.
- ✓ The C-663 is set for open-loop operation.  
For details on setting manually, see "Setting the Operating Mode" (p. 82).
- ✓ You have done a reference move for the axis.

If you work with PIMikroMove: You have done steps 1 to 4 of the instructions "Starting Motion" (p. 71).

### Open-loop operation with sensor: Position adjustment

1. If you use PIMikroMove, open the window for sending commands:
  - a) When the **Start up controller** window is open, close it by clicking **Close** or **Cancel** to go to the main window.
  - b) Select the **Tools > Command entry** menu item in the main window or press **F4** on the keyboard.
- In PITerminal the main window from which commands can be sent is opened automatically after establishing communication.
2. Query the axis position by sending **POS?**.
3. Deactivate the motor by sending **SVO <AxisID> 0**, whereby <AxisID> identifies the axis.
4. Send **TSP 1 <Position>** to set the position measured by the sensor to the position value that was queried with **POS?**. <Position> denotes the position value to be set.
5. Activate the motor by sending **SVO <AxisID> 1**, whereby <AxisID> identifies the axis.
6. Send **POS?** and **TSP?** consecutively to retrieve the current axis position and the sensor position. The positional values should be about the same.

# 7 Operation

## In this Chapter

Protective Functions of the C-663 .....	85
Data Recorder .....	87
Digital Output Signals.....	89
Digital Input Signals.....	96
Analog Input Signals.....	101
Joystick Control.....	102
Controller Macros .....	110

### 7.1 Protective Functions of the C-663

#### 7.1.1 Behavior During System Errors

There is a system error when C-663 is not responsive.

For example, the cause of a system error can be a buffer overflow in the firmware of the C-663.

If a system error occurs the C-663 reacts as follows:

- After a certain delay, the safety function of the **Watchdog Timer** initiates a reboot of the C-663.

#### 7.1.2 Behavior with Motion Errors

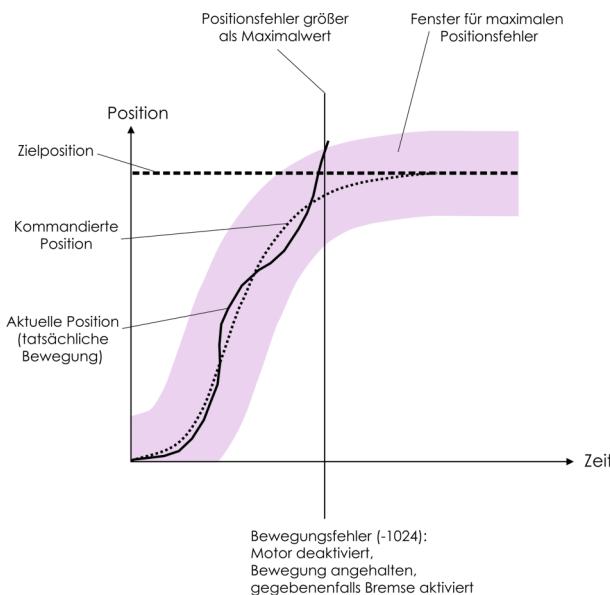
Motion errors can be caused for example, by malfunctions of the drive or the position sensor of the positioner.

A motion error occurs, when the position error (i.e., the absolute value of the difference between the current position and the commanded position) exceeds the specified maximum value in closed-loop operation. The range in which the deviation may lie is specified by the **Maximum Position Error (Phys. Unit)** parameter (ID 0x8).

If motion error occurs, the C-663 reacts as follows to protect the system against damage:

- The motor is deactivated for the affected axis.
- If applicable, the brake is activated for the axis in question.
- All motion is stopped.
- Error code -1024 is set.

Then restore the operational readiness (p. 86) for the C-663.



### INFORMATION

With the `CTO` (p. 149) and `TRO` (p. 211) commands, you can program the digital output lines of the C-663 so that they are activated in the case of motion errors. The programmed output lines remain activated until the error code is reset to 0. Refer to "Configuring the "Motion Error" Trigger Mode" (p. 93) for details.

## 7.1.3 Restoring Operational Readiness

### NOTICE



#### Unintentional motion after brake deactivated!

If servo mode is switched off, e.g., after a motion error occurs, the brake of the positioner can be deactivated by command. Deactivating the brake can cause the positioner to move unintentionally.

- Secure the positioner against moving unintentionally before you deactivate the brake by command!

#### Restoring operational readiness

1. Send the `ERR?` command to read out the error code.  
`ERR?` resets the error code to zero during the query.
2. Check your system and make sure that all axes can be moved safely.

3. If the motor has been deactivated after an error:

- Activate the motor for the axis with the `SVO` command.

When the motor is activated, the target position is set to the current axis position and the brake is deactivated if applicable. Now the axis can move again and you can command a new target position.

## 7.2 Data Recorder

### 7.2.1 Configuring the Data Recorder

The C-663 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-663. After the C-663 has been switched on or rebooted, factory settings will be active unless a configuration is done with a startup macro.

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

- Send the `HDR?` command (p. 170).

The options available for recording and triggering are displayed together with 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. 160) 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-663 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. 158).

### Configuring the recording trigger

You can specify how the recording is to be triggered.

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

### Setting the record table rate

- Send the `RTR?` command (p. 198) to read out the record table rate of the data recorder.  
The parameter indicates the number of servo cycles required for recording each data point. The default value is 10 servo cycles. The servo cycle time of the C-663 is 50 µs.
- Change the record table rate with the `RTR` command (p. 198).

As the record table rate increases, the maximum duration of the data recording is increased.

## 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. 206).

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. 161).  
The data is output in the GCS array format (refer to the SM146E user manual).
- Query the number of points in the last recording with the `DRL?` command (p. 160).

## 7.3 Digital Output Signals

The digital outputs of the C-663 are available at the **I/O** socket (p. 281).

- Get the number of the output lines available on the C-663 with the **TIO?** command (p. 209).

External devices can be triggered via the digital outputs of the C-663. Potential applications:

- Linking the trigger output to the motion of the axis. Details and examples can be found in this section.
- Direct switching of output lines, e. g., in macros. Details and examples of macros can be found in "Controller Macros" (p. 110).

### 7.3.1 Commands for Digital Outputs

The following commands are available for the use of digital outputs:

Command	Syntax	Function
CTO	CTO {<TrigOutID> <CTOPam> <Value>}	Configures the conditions for the trigger output. Couples the trigger output to the axis motion.
DIO	DIO {<DIOID> <OutputOn>}	Switches digital output lines directly to the low or high state, either separately or all lines at once. Should not be used for output lines where the trigger output is activated by <b>TRO</b> .
TRO	TRO {<TrigOutID> <TrigMode>}	Activates or deactivates the trigger output conditions set with <b>CTO</b> . Default: Trigger output deactivated.

One configuration setting can be made per **CTO** command:

**CTO <TrigOutID> <CTOPam> <Value>**

- **<TrigOutID>** is one digital output line of the controller.
- **<CTOPam>** is the CTO parameter ID in decimal format.
- **<Value>** is the value to which the CTO parameter is set.

The following trigger modes (**<Value>**) can be set for **<CTOPam> = 3**:

<Value>	Trigger mode	Short description
0 (default)	Position Distance	Once the axis has moved a specified distance, a trigger pulse is output (p. 90). Optionally, start and stop values can be defined to limit triggering to one position range and one particular direction of motion (negative or positive).

<Value>	Trigger mode	Short description
2	On Target	The on-target state of the axis selected is output at the selected trigger output (p. 92).
5	Motion Error	The selected digital output line becomes active when a motion error occurs (p. 93). The line stays active until the error code is reset to 0 (by an <code>ERR?</code> query).
6	In Motion	The selected digital output line is active as long as the selected axis is in motion (p. 93).
7	Position+Offset	The first trigger pulse is output when the axis has reached a specified position. The next trigger pulses are each output when the axis position equals the sum of the last valid trigger position and a specified distance. The trigger output is stopped when a stop value is reached. The polarity sign of the distance value determines the direction of motion in which trigger pulses are to be output. Refer to "Configuring the "Position + Offset" Trigger Mode" (p. 94).
8	Single Position	The selected digital output line is active when the axis position has reached or exceeded a specified position (p. 95).

In addition, the polarity (active high / active low) of the signal at the digital output can be set (p. 96).

#### INFORMATION

The settings for the configuration of the digital output lines can only be modified in the volatile memory of the C-663. After switching on or restarting the C-663 the factory default settings are active, unless a configuration has already been made using the startup macro.

### 7.3.2 Configuring the "Position Distance" Trigger Mode

The *Position Distance* trigger mode is suitable for scanning applications. Once the axis has moved along the distance that was set with CTO parameter ID = 1 (TriggerStep), a trigger pulse is output. The pulse width is one servo cycle.

The unit of the distance (TriggerStep) is subject to the settings of parameters 0xE and 0xF. Default is mm.

1. Configure the digital output line (<TrigOutID>) that is to be used as the trigger output:
  - Send `CTO <TrigOutID> 2 A`, where *A* indicates the axis to be moved.
  - Send `CTO <TrigOutID> 3 0`, where 0 specifies the *Position Distance* trigger mode.
  - Send `CTO <TrigOutID> 1 S`, where *S* indicates the distance.
2. If you want to activate the conditions for trigger output, send `TRO <TrigOutID> 1`.

**Example:**

A pulse on digital output line 1 is output every time the axis 1 of the positioner has covered a distance of 0.1 µm.

- Send:

```
CTO 1 2 1  
CTO 1 3 0  
CTO 1 1 0.0001  
TRO 1 1
```

**"Position Distance" trigger mode with start and stop values for positive motion direction of the axis**

Optionally, you can define start and stop values for limiting the range and for specifying the motion direction of the axis (positive or negative).

**INFORMATION**

If start and stop values have the same value, they are ignored.

If the direction of motion is reversed before the axis position has reached the stop value, trigger pulses continue to be output.

1. Configure the digital output line (<TrigOutID>) that is to be used as the trigger output:
  - Send `CTO <TrigOutID> 2 A`, where *A* indicates the axis to be moved.
  - Send `CTO <TrigOutID> 3 0`, where 0 specifies the *Position Distance* trigger mode.
  - Send `CTO <TrigOutID> 1 S`, where *S* indicates the distance.
  - Send `CTO <TrigOutID> 8 Start`, where *Start* indicates the start value.
  - Send `CTO <TrigOutID> 9 Stop`, where *Stop* indicates the stop value.
2. If you want to activate the conditions for trigger output, send `TRO <TrigOutID> 1`.

**Example**

A pulse on digital output line 1 is output every time the axis 1 of the positioner has covered a distance of 0.1 µm, as long as axis 1 is moving in positive direction of motion within the range of 0.2 µm to 0.55 µm (start value < stop value).

- Send:

```
CTO 1 2 1  
CTO 1 3 0  
CTO 1 1 0.0001
```

```
CTO 1 8 0.0002
CTO 1 9 0.00055
TRO 1 1
```

**"Position Distance" trigger mode with start and stop values for negative motion direction of the axis**

The above example is presented with interchanged start and stop values in the following. Triggering occurs in negative motion direction of the axis (stop value < start value) in the range between 0.55 µm and 0.2 µm.

**Example:**

➤ Send:

```
CTO 1 2 1
CTO 1 3 0
CTO 1 1 0.0001
CTO 1 8 0.00055
CTO 1 9 0.0002
TRO 1 1
```

### 7.3.3 Configuring the "On Target" Trigger Mode

The on-target state of the axis selected (p. 32) is output at the selected trigger output in *On Target* trigger mode.

1. Configure the digital output line (<TrigOutID>) to be used as the trigger output:
  - Send `CTO <TrigOutID> 2 A`, where *A* indicates the axis to be moved.
  - Send `CTO <TrigOutID> 3 2`, where 2 specifies the *On Target* trigger mode.
2. If you want to activate the conditions for trigger output, send `TRO <TrigOutID> 1`.

**Example:**

The on-target state of axis 1 is to be output on the digital output line 1.

➤ Send:

```
CTO 1 2 1
CTO 1 3 2
TRO 1 1
```

### 7.3.4 Configuring the "Motion Error" Trigger Mode

The *Motion Error* trigger mode is suitable for monitoring motion. The selected digital output line becomes active when a motion error occurs on one of the connected axes. The line stays active until the error code is reset to 0 (by an `ERR?` query).

#### INFORMATION

A motion error occurs when the current position differs too much from the commanded position during motion.

For further information, see "Motion Error".

1. Configure the digital output line (`<TrigOutID>`) that is to be used as the trigger output:
  - Send `CTO <TrigOutID> 3 5`, where 5 specifies the *Motion Error* trigger mode.
2. If you want to activate the conditions for trigger output, send `TRO <TrigOutID> 1`.

### 7.3.5 Configuring the "In Motion" Trigger Mode

The motion state of the selected axis is output at the selected trigger output in *In Motion* trigger mode. The line is active, as long as the selected axis is in motion.

The motion state can also be read with the `#5` (p. 139), `#4` (p. 138), and `SRG?` (p. 204) commands.

#### INFORMATION

If the axis is in motion, then bit 14 of the state register 1 of the axis is set.

1. Configure the digital output line (`<TrigOutID>`) that is to be used as the trigger output:
  - Send `CTO <TrigOutID> 2 A`, where A indicates the axis to be moved.
  - Send `CTO <TrigOutID> 3 6`, where 6 specifies the *In Motion* trigger mode.
2. If you want to activate the conditions for trigger output, send `TRO <TrigOutID> 1`.

#### Example:

Digital output line 1 is to be active if axis 1 of the positioner is in motion.

➤ Send:

`CTO 1 2 1`

`CTO 1 3 6`

`TRO 1 1`

### 7.3.6 Configuring the "Position + Offset" Trigger Mode

The *Position+Offset* trigger mode is suitable for scanning applications. The first trigger pulse is output when the axis has reached a specified position (*TriggerPosition*). The next trigger pulses are output respectively when the axis position equals the sum of the last valid trigger position and a specified distance (*TriggerStep*). The trigger output is stopped when a stop value is reached. The polarity sign of the distance value determines the direction of motion in which trigger pulses are to be output.

The pulse width is one servo cycle.

The unit for *TriggerPosition*, *TriggerStep* and stop value is subject to the settings of parameters 0xE and 0xF. Default is mm.

1. Configure the digital output line (<TrigOutID>) that is to be used as the trigger output:
  - Send `CTO <TrigOutID> 2 A`, where *A* indicates the axis to be moved.
  - Send `CTO <TrigOutID> 3 7`, where 7 specifies the *Position+Offset* trigger mode.
  - Send `CTO <TrigOutID> 1 S`, where *S* indicates the distance.
  - Send `CTO <TrigOutID> 10 TriPos`, where *TriPos* indicates the position for the output of the first trigger pulse.
  - Send `CTO <TrigOutID> 9 Stop`, where *Stop* indicates the stop value.
2. If you want to activate the conditions for trigger output, send `TRO <TrigOutID> 1`.

#### Example 1:

The first trigger pulse is to be output on digital output line 1 if the absolute position of axis 1 is 1.5 mm. A pulse should then be output on this line every time axis 1 has covered a distance of 0.1 µm in the positive direction. The last trigger pulse is to be output if the absolute axis position is 2.5 mm.

- Send:

```
CTO 1 2 1
CTO 1 3 7
CTO 1 1 0.0001
CTO 1 10 1.5
CTO 1 9 2.5
TRO 1 1
```

#### Example 2:

The first trigger pulse is to be output on digital output line 2 if the absolute position of axis B is 0.4 mm. A pulse should then be output on this line every time axis B has covered a distance of

1 µm in the negative direction. The last trigger pulse is to be output if the absolute axis position is 0.1 mm.

➤ Send:

```
CTO 2 2 B
CTO 2 3 7
CTO 2 1 -0.001
CTO 2 10 0.4
CTO 2 9 0.1
```

#### INFORMATION

The velocity setting of the axis must be appropriate for the distance setting (TriggerStep) commanded by the CTO command. Recommended value:

Maximum velocity = distance \* 20 kHz / 2

where 20 kHz is the servo cycle frequency of the C-663.

### 7.3.7 Configuring the "Single Position" Trigger Mode

The selected digital output line is active in *Single Position* trigger mode, when the axis position has reached or exceeded a specified position (TriggerPosition).

The unit for TriggerPosition depends on the settings of the parameters 0xE and 0xF. Default is mm.

1. Configure the digital output line (<TrigOutID>) to be used as the trigger output:
  - Send CTO <TrigOutID> 2 A, where A indicates the axis to be moved.
  - Send CTO <TrigOutID> 3 8, where 8 specifies the *Single Position* trigger mode.
  - Send CTO <TrigOutID> 10 TriPos, where *TriPos* indicates the position at which the output line is to become active.
2. If you want to activates the conditions for trigger output, send TRO <TrigOutID> 1.

#### Example:

Digital output line 1 is to become active when the absolute position of axis 1 is at least 1.5 mm.

➤ Send:

```
CTO 1 2 1
CTO 1 3 8
CTO 1 10 1.5
```

### 7.3.8 Setting Signal Polarity

The polarity of the signal at the digital output which is used for triggering can be selected with the *Polarity* CTO parameter. The polarity can have the following values:

- active high = 1 (default setting)
- active low = 0
- Configure the digital output line (<TrigOutID>) to be used as the trigger output:
  - Send `CTO <TrigOutID> 7 P`, where *P* indicates the polarity.

Example:

The signal polarity for digital output line 1 is to be set to active low.

- Send:

```
CTO 1 7 0
```

## 7.4 Digital Input Signals

The digital inputs of the C-663 are available on the **I/O** socket (p. 281).

- Get the number of the input lines available on the C-663 with the `TIO?` command (p. 209).
- Get the state of the input lines with the `DIO?` command (p. 158).

Potential applications:

- Use in macros (p. 99). Details and examples of macros can be found in "Controller Macros" (p. 110).
- Use as switch signals (p. 99)

#### INFORMATION

The digital inputs (pins 1 to 4) on the **I/O** socket can also be used as analog inputs.

- Digital: TTL
- Analog: 0 to +5 V

## 7.4.1 Commands and Parameters for Digital Inputs

### Commands

The following commands are available for the use of digital inputs:

Command	Syntax	Function
CPY	CPY <Variable> <CMD?>	Copies the state of a digital input line to a variable when used in conjunction with the DIO? query command. Use in macros to set local variables (p. 132).
DIO?	DIO? [{<DIOID>}]	Gets the state of the digital input lines.
FED	FED {<AxisID> <EdgeID> <Param>}	Starts a move to a signal edge. The signal source can be a digital input line.
FNL	FNL [{<AxisID>}]	Starts a reference move to the negative physical limit of the travel range A digital input line can be used as the source of the negative limit switch signal instead of the negative limit switch.
FPL	FPL [{<AxisID>}]	Starts a reference move to the positive physical limit of the travel range Limit switch. A digital input line can be used as the source of the positive limit switch signal instead of the positive limit switch.
FRF	FRF [{<AxisID>}]	Starts a reference move to the reference switch. A digital input line can be used as the source of the reference switch signal instead of the reference switch.
JRC	JRC <Jump> <CMD?> <OP> <Value>	Can only be used in macros. Triggers a relative jump of the macro run pointer depending on the state of a digital input line when used in conjunction with the DIO? query command.
MEX	MEX <CMD?> <OP> <Value>	Can only be used in macros. Stops running of the macro depending on the state of a digital input line when used in conjunction with the DIO? query command.
WAC	WAC <CMD?> <OP> <Value>	Can only be used in macros. Waits until a digital input line reaches a certain state when used in conjunction with the DIO? query command.

## Parameters

The following parameters are available for the configuration of digital inputs:

Parameters	Description and Possible Values
<b>Source Of Reference Signal</b> 0x5C	Specifies the source of the reference signal for the <b>FRF</b> and <b>FED</b> commands: 0 = reference switch 1 = Digital input 1 2 = Digital input 2 3 = Digital input 3 4 = Digital input 4
<b>Source Of Negative Limit Signal</b> 0x5D	Specifies the source(s) of the negative limit switch signal for the <b>FNL</b> and <b>FED</b> commands via a bitmask: 0 = Negative limit switch (default setting) 1 = Digital input 1 (bit 0) 2 = Digital input 2 (bit 1) 4 = Digital input 3 (bit 2) 8 = Digital input 4 (bit 3)
<b>Source Of Positive Limit Signal</b> 0x5E	Specifies the source(s) of the positive limit switch signal for the <b>FPL</b> and <b>FED</b> commands via a bitmask: 0 = Positive limit switch (default setting) 1 = Digital input 1 (bit 0) 2 = Digital input 2 (bit 1) 4 = Digital input 3 (bit 2) 8 = Digital input 4 (bit 3)
<b>Invert Digital Input Used For Negative Limit</b> 0x5F	Inverts the polarity of the digital inputs, which are used for the source of the negative limit switch signal, via a bitmask: 0 = No digital input inverted (default setting). 1 = Digital input 1 inverted (bit 0) 2 = Digital input 2 inverted (bit 1) 4 = Digital input 3 inverted (bit 2) 8 = Digital input 4 inverted (bit 3)
<b>Invert Digital Input Used For Positive Limit</b> 0x60	Inverts the polarity of the digital inputs, which are used for the source of the positive limit switch signal, via a bitmask: 0 = No digital input inverted (default setting). 1 = Digital input 1 inverted (bit 0) 2 = Digital input 2 inverted (bit 1) 4 = Digital input 3 inverted (bit 2) 8 = Digital input 4 inverted (bit 3)

## 7.4.2 Using Digital Input Signals in Macros

The digital inputs on the **I/O** socket can be used in macros as follows:

- Conditional running of the macro
- Conditional stopping of the macro
- Conditional jump of the macro pointer
- Copying the input state to a variable

Further information and examples can be found in "Controller Macros" (p. 110).

### INFORMATION

You can connect the C-170.PB pushbutton box from PI to the **I/O** socket (p. 281) to generate the digital input signals for use in macros. It also displays the state of the digital output lines via LEDs.

## 7.4.3 Using Digital Input Signals as Switch Signals

The digital inputs on the **I/O** socket can be used as the source of reference point and limit switch signals (e.g., for reference moves (p. 38)) for an axis.

### Using digital input as reference signal

### INFORMATION

The level of the digital input signal which you use instead of the reference switch may only change once across the entire travel range.

- Use a suitable signal source.
- If necessary, invert the signal logic of the digital input line by setting the **Invert Reference?** parameter (ID 0x31) accordingly.

### INFORMATION

The **Has Reference?** parameter (ID 0x14) has no influence on the use of a digital input line as the source of the reference signal.

- Select the source of the reference signal for the axis by changing the **Source Of Reference Signal** parameter (ID 0x5C).

Detailed information on changing parameters can be found in "Adapting Settings" (p. 245).

## Using digital inputs as source of the limit switch signals

### INFORMATION

Several digital inputs can be selected as the source for a limit switch signal.

If a limit switch signal is used for reference moves, only one digital input line may be selected as the source of the limit switch signal.

### INFORMATION

The level of the digital input signal which you use instead of an internal limit switch may only change once across the entire travel range.

- Use suitable signal sources.
- If necessary, invert the signal logic of the digital input lines by setting parameters **Invert Digital Input Used For Negative Limit** (ID 0x5F) and **Invert Digital Input Used For Positive Limit** (ID 0x60) accordingly.

### INFORMATION

The **Has No Limit Switches?** parameter (ID 0x32) determines whether the C-663 evaluates the signals from the internal limit switches of the positioner. This parameter has no influence on the use of digital input lines as the source of the limit switch signal.

- Select the source(s) of the negative limit switch signal for the axis by changing the **Source Of Negative Limit Signal** parameter (ID 0x5D).
- Select the source(s) of the positive limit switch signal for the axis by changing the **Source Of Positive Limit Signal** parameter (ID 0x5E).

Detailed information on changing parameters can be found in "Adapting Settings" (p. 245).

#### Example:

Digital input lines 1, 3, and 4 are to be used for axis 1 as the sources of the positive limit switch signal. In addition, the signal polarity of lines 1 and 3 is to be inverted for axis 1. All adaptations are made in the volatile memory of the C-663 only.

- Send:

SPA 1 0x5E 13, to select lines 1, 3, and 4.

SPA 1 0x60 5, to invert the signal polarity of lines 1 and 3.

## 7.5 Analog Input Signals

The analog inputs of the C-663 are available on the **I/O** socket (p. 281).

- Get the number of the analog input lines available on the C-663 with the **TAC?** command (p. 208).
- Query the voltage on the analog inputs with the **TAV?** command (p. 209).
- Use the data recorder (p. 87) to record the analog input signals.

Potential applications:

- Use in macros (p. 102): Details and examples of macros are found in "Controller Macros" (p. 110).
- Scanning applications with PIMikroMove (see PIMikroMove manual)

### **INFORMATION**

The analog inputs (pins 1 to 4) on the **I/O** socket can also be used as digital inputs.

- Analog: 0 to +5 V
- Digital: TTL

### 7.5.1 Commands for Analog Inputs

The following commands are available for the use of analog inputs:

Command	Syntax	Function
<b>CPY</b>	CPY <Variable> <CMD?>	Copies the voltage value of an analog input line to a variable when used in combination with the TAV? query command. Use in macros to set local variables (p. 132).
<b>DRC</b>	DRC {<RecTableID> <Source> <RecOption>}	Configures the data recorder. Analog input values can be recorded using record option 81.
<b>JRC</b>	JRC <Jump> <CMD?> <OP> <Value>	Can only be used in macros. Triggers a relative jump of the pointer when running the macro depending on the voltage at an analog input line when used in conjunction with the TAV? query command.
<b>MEX</b>	MEX <CMD?> <OP> <Value>	Can only be used in macros. Stops running of the macro depending on the voltage at an analog input line when used in conjunction with the TAV? query command.
<b>TAC?</b>	TAC?	Get the number of installed analog lines.
<b>TAV?</b>	TAV? [<AnalogInputID>]	Get voltage at analog input.

Command	Syntax	Function
WAC	WAC <CMD?> <OP> <Value>	Can only be used in macros. Waits until an analog input line reaches a certain voltage when used in conjunction with the TAV? query command.

## 7.5.2 Using Analog Input Signals in Macros

The analog inputs on the I/O socket can be used in macros as follows:

- Conditional running of the macro
- Conditional stopping of the macro
- Conditional jump of the macro pointer
- Copying the input state to a variable

Further information and examples can be found in "Controller Macros" (p. 110).

## 7.6 Joystick Control

### 7.6.1 How Joystick Control Works

The joystick axis controls the velocity of the positioner axis connected to the C-663 (commanded velocity output from the profile generator).

The relationship between the displacement of the joystick axis and the velocity of the positioner axis is established by the C-663 using a lookup table. The 256 values in the lookup table are factors that are applied to the velocity set with the VEL command (p. 217) during joystick control. The value range is from -1.0000 to 1.0000.

The firmware of the controller provides two predefined lookup table types to choose from (linear and parabolic) and allows the lookup table to be filled with custom values. The content of the lookup table is automatically saved in the nonvolatile memory of the C-663.

During joystick control, the soft limit specified by the parameter 0x15 or 0x30 is used as the target position. For details, see "Travel Range and Soft Limits" (p. 34). When disabling the joystick, the current position of the joystick-controlled axis is set as the new target position.

#### INFORMATION

Motion commands are not allowed when a joystick is activated for the axis.

When the motor is deactivated, no joystick control is possible.

## 7.6.2 Commands and Parameters for Joystick Control

### Commands

The following commands are available for the use of the joystick:

Command	Syntax	Function
JON	JON {<JoystickID> <uint>}	Activates or deactivates a joystick connected to the controller.
JON?	JON? [{<JoystickID>}]	Gets the activation state of the joystick.
JAX	JAX <JoystickID> <JoystickAxis> <AxisID>	Specifies the axis that is controlled by a joystick connected to the controller.
JAX?	JAX? [{<JoystickID> <JoystickAxis>}]	Gets the axis that is controlled by a joystick connected to the controller.
JAS?	JAS? [{<JoystickID> <JoystickAxis>}]	Gets the current state of a joystick axis (displacement).
JBS?	JBS? [{<JoystickID> <JoystickButton>}]	Gets the current state of a joystick button (pressed or not pressed).
JDT	JDT {<JoystickID> <JoystickAxis> <uint>}	Specifies a default lookup table type for a joystick axis.
JLT	JLT <JoystickID> <JoystickAxis> <Addr> <floatn>	Fills the lookup table for a joystick axis with custom values.
JLT?	JLT? [<StartPoint> <NumberOfPoints> [{<JoystickID> <JoystickAxis>}]]	Gets the current valid lookup table values.

### Parameters

The following parameters are available for the use of the joystick:

Parameters	Description and possible values
<b>Invert Direction Of Motion For Joystick-Controlled Axis?</b> 0x61	Specifies the direction of motion for joystick-controlled axes. 0 = direction of motion not inverted (default setting) 1 = direction of motion inverted

### 7.6.3 Controlling Axis Motion

PIMikroMove is used in the following to activate joystick control for the positioner. It is not necessary to know the corresponding GCS commands.

#### NOTICE



##### Unintentional motion while activating the joystick!

If no joystick is connected to the C-663, activating the joystick in the software can cause unintentional motion of the axis connected.

- Activate the joystick in the software only if a joystick is actually connected to the C-663.

#### INFORMATION

The use of macros provides a wide range of application possibilities for joystick control. In particular, the joystick button can be used in macros for a wide variety of applications. Details and examples of macros are found in "Controller Macros" (p. 110).

#### INFORMATION

The C-663 supports one logical axis and is therefore normally used with positioners that only have one motion axis. In this case, the designation "axis" is synonymous with "positioner". Therefore, no distinction is made between "positioner" and "axis" in the following operational instructions.

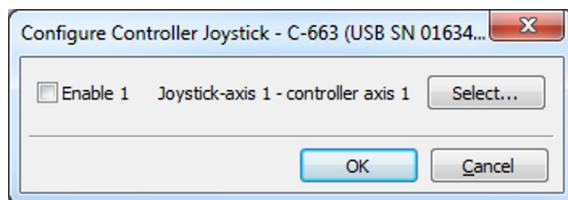
#### Requirement

- ✓ You have started operation of the positioner (p. 71) and initial motion with PIMikroMove.
- ✓ You have connected a joystick to the C-663 (p. 54).

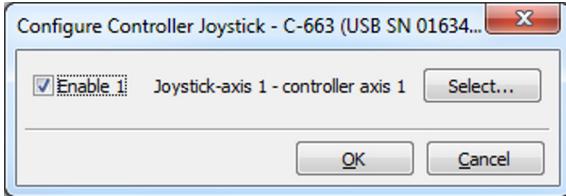
#### Controlling axis motion via a joystick

1. Open the **Configure Controller Joystick** window in the main window of PIMikroMove via the **C-663 > Configure controller joystick(s)...** menu item.

The C-663 joystick and its axis are listed in the **Configure Controller Joystick** window.



2. Assign the axis to be moved to the joystick axis:
  - a) Click **Select** in the **Configure Controller Joystick** window.
  - b) Mark the correct positioner name in the **Select controller axis** window.

- c) Click **OK** in the **Select controller axis** window to confirm selection and close the window.
3. Activate the joystick in the **Configure Controller Joystick** window by checking the respective **Enable** checkbox.
- 
- If joystick control does not operate satisfactorily or the positioner moves even though you are not moving the joystick:
- Check whether the joystick is locked mechanically.
  - Calibrate the joystick (p. 105).
4. Control the velocity of the positioner via the joystick.
5. If you want to deactivate joystick control, remove the tick from the respective **Enable** checkbox in the **Configure Controller Joystick** window.

#### 7.6.4 Calibrating the Joystick

Calibration of the individual joystick axes is recommended after connecting a joystick to the C-663 for the first time.

Calibration involves the following steps:

- If corresponding operating elements are on the joystick: Adjusting the joystick axes mechanically.
- Calibrating the joystick axes in PIMikroMove

#### INFORMATION

The lookup table type to be used is selected or the lookup table is filled with custom values when calibrating the joystick in PIMikroMove. To do this, a positioner does not need to be connected to the C-663.

Calibration is necessary for proper joystick control in the following cases:

- After activating joystick control, the positioner moves even though you are not moving the joystick.
- The response behavior of the joystick does not correspond to your requirements.
- You are using the Z axis of a C-819.30 joystick.

**INFORMATION**

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

- Calibrate the joystick axes in PIMikroMove only if it is necessary.
- Contact our customer service department (p. 273) if the C-663 shows unexpected behavior.

**INFORMATION**

It is not possible to adjust the Z axis of C-819.30's joystick mechanically and it cannot be operated using the C-663's standard lookup table types (linear or parabolic).

- Calibrate the Z axis of the joystick after connecting to the C-663 with PIMikroMove.
- Use the **Measure Joystick Parameters and Use Custom Lookup Table** method for calibrating the Z axis.
- Repeat calibration of the Z axis if you are connecting the Z axis to another controller.

**INFORMATION**

The parabolic lookup table type allows greater sensitivity at low velocity.

**Adjusting a joystick axis mechanically**

- Check whether the positioner moves when joystick control is activated even though you are not moving the joystick.

If so:

- Check whether the joystick is locked mechanically and unlock if necessary.
- Keep the affected joystick axis (that means for example, the control lever) at the center position and adjust it with the appropriate operating elements until the positioner no longer moves. With the C-819.20 and C-819.30 joysticks, turn the corresponding rotary knob for adjustment (p. 108).

If not:

- Check whether the response behavior of the joystick corresponds to your requirements.

If so:

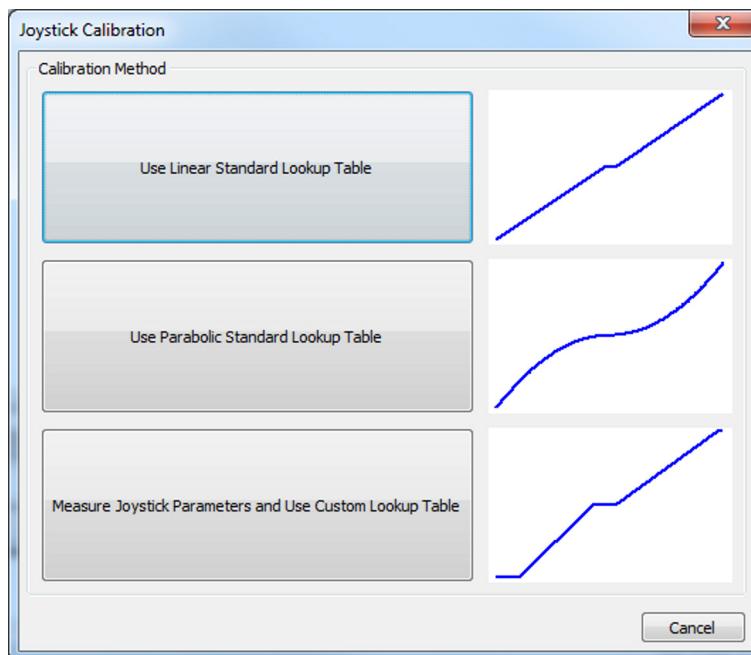
- The calibration is finished.

If not:

- Calibrate the joystick axis in PIMikroMove.

### Calibrating the joystick axis in PIMikroMove

1. Open the **Joystick Calibration** window in the main window of PIMikroMove via the **C-663 > Calibrate controller joystick...** menu item.



2. Select the calibration method by clicking on the appropriate button:
  - If you want to use the linear lookup table type for the joystick axis, click **Use Linear Standard Lookup Table**. With this, the appropriate lookup table type is loaded and the calibration is finished.
  - If you want to use the parabolic lookup table type for the joystick axis, click **Use Parabolic Standard Lookup Table**. With this, the appropriate lookup table type is loaded and the calibration is finished.
  - If you have connected the Z axis of a C-819.30 joystick or generally want to map the behavior of the joystick in an individual lookup table, click **Measure Joystick Parameters and Use Custom Lookup Table**. Here the **Controller Joystick Calibration** window opens.
3. When the **Controller Joystick Calibration** window has opened, follow the instructions in this window.

The custom lookup table values are determined in this way.

By clicking on **OK**, you load the lookup table values into the nonvolatile memory of the C-663. Calibration is now finished.

### 7.6.5 Joysticks Available

PI provides the joysticks described in the following as optional accessories (p. 12).

#### Analog C-819.20 joystick, 2 axes

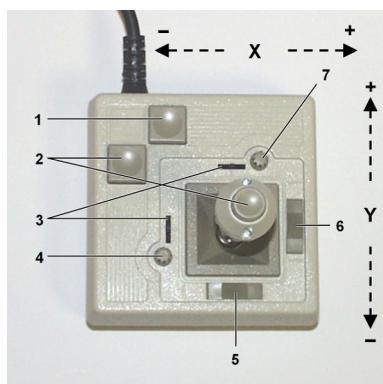


Figure 16: C-819.20 joystick

- 1 Pushbutton for the X axis
- 2 Pushbutton for the Y axis
- 3 Adjustment indicator
- 4 Rotary knob for adjustment of the Y axis (calibration)
- 5 X axis lock
- 6 Y axis lock
- 7 Rotary knob for adjustment of the X axis (calibration)

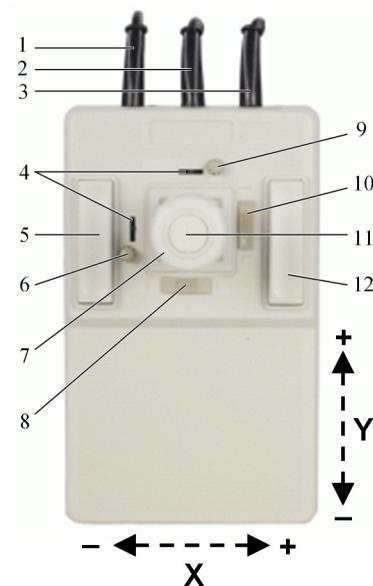
**C-819.30 analog joystick, 3 axes**

Figure 17: C-819.30 joystick

- 1 Cable for the Z axis
- 2 Cable for the Y axis
- 3 Cable for the X axis
- 4 Adjustment indicator
- 5 Pushbutton for the Y axis
- 6 Rotary knob for adjustment of the Y axis (calibration)
- 7 XY control lever with rotary knob for Z axis
- 8 X axis lock
- 9 Rotary knob for adjustment of the X axis (calibration)
- 10 Y axis lock
- 11 Pushbutton for the Z axis
- 12 Pushbutton for the X axis



Figure 18: C-819.30 joystick, rotary knob for the Z axis

## 7.7 Controller Macros

### 7.7.1 Overview: Macro Functionality and Example Macros

The C-663 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-663 is switched on or rebooted.
- Processing and stopping a macro can be linked to conditions. This makes loops possible.
- Macros can call up themselves or other macros.
- Variables (p. 132) can be set for the macro and in the macro itself and used in different operations.
- Input signals can be evaluated for conditions and variables.

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

- Moving an axis back and forth (p. 114)
- Recording a macro for a controller whose address is different to 1 (p. 115)
- Moving an axis with a variable travel range back and forth (p. 117)
- Implementing multiple calls of a macro via a loop (p. 118)
- Preparing an axis for operation via a startup macro (p. 119)
- Synchronizing two controllers (p. 121)
- Stopping motion by pushbutton (p. 122)
- Joystick control with storage of positions (p. 123)
- Joystick control with change in velocity (p. 127)

## 7.7.2 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. 142)	ADD <Variable> <FLOAT1> <FLOAT2>	Adds two values and saves the result to a variable (p. 132). Can only be used for local variables in macros.
CPY (p. 147)	CPY <Variable> <CMD?>	Copies a command response to a variable (p. 132). Can only be used for local variables in macros.
DEL (p. 154)	DEL <uint>	Can only be used in macros. Delays <uint> milliseconds.
JRC (p. 182)	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. 184)	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. 186)	Lists all macros or the content of a specified macro.
MEX (p. 189)	MEX <CMD?> <OP> <Value>	Can only be used in macros. Stops the macro execution depending on a condition.
RMC? (p. 195)	RMC?	Lists macros which are currently running.
VAR (p. 215)	VAR <Variable> <String>	Sets a variable (p. 132) to a certain value or deletes it. Can only be used for local variables in macros.

Command	Syntax	Function
VAR? (p. 216)	VAR? [{<Variable>}]	Gets variable values.
WAC (p. 218)	WAC <CMD?> <OP> <Value>	Can only be used in macros. Waits until a condition is met.
#8 (p. 140)	-	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>

## 7.7.3 Working with Macros

Work with macros comprises the following:

- Recording macros (p. 112)
- Starting macros (p. 116)
- Stopping macros (p. 118)
- Configuring a startup macro (p. 119)
- Deleting macros (p. 120)

### INFORMATION

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

### Recording a macro

### INFORMATION

The C-663 can save up to 32 macros simultaneously. A maximum of 5 nesting levels are possible in macros.

**INFORMATION**

Basically all GCS commands (p. 129) can be included in a macro. Exceptions:

- **RBT** for rebooting the C-663
- **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**

If you record a macro on a C-663 whose controller address is different to 1, note the following when entering the commands that are to be an element of the macro:

- If you are working with PIterminal and have established communication with the **Connect...** button, the target address has to be typed in in every command line.
- If you are working with PIMikroMove or have established communication with PIterminal using the **GCS DLL...** button, the target address is automatically sent and may not be typed in.

**INFORMATION**

To make the use of macros more flexible, you can use local and global variables in macros. For further information, see "Variables" (p. 132).

**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. 132) in macros to make macros more flexible, and give the corresponding variable values when starting macro execution.
- Contact our customer service department (p. 273) if the C-663 shows unexpected behavior.

**INFORMATION**

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

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-663.
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-663 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 `MAC BEG` and `MAC END` commands 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 macrol1
MAC START macro2
MAC END

```

**Example: Recording macro for controller whose address is different to 1**

**INFORMATION**

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

The controller address is set to 2 via the DIP switches. In this example, macro recording is done using PITerminal, whereby communication was established with the **Connect...** button (as a result, the target address has to be typed in every command line).

The motor is to be activated for axis 1 via the REF macro and a reference move to the reference switch is started.

1. Record the macro by sending:

```

2 MAC BEG ref
2 SVO 1 1
2 DEL 1000
2 FRF 1
2 MAC END

```

2. Check the content of the ref macro by sending:

```
2 MAC? ref
```

The response reads:

```

0 2 SVO 1 1
DEL 1000
FRF 1

```

The first line of the response contains the target and sender address corresponding to the GCS syntax for multiline responses. However, the target address is not included in the macro.

### 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. 245).

2. Start the macro:

- If the macro is to be run once, send the `MAC START macroname string` command, whereby *macroname* indicates the name of the macro.
- If the macro is to be run *n* times, send the `MAC NSTART macroname n string` command, whereby *macroname* indicates the name of the macro and *n* indicates the number of runs.

*string* stands for the values of local variables. The values only have to be specified when the macro contains corresponding local variables. The sequence of the values in the input must correspond to the numbering of the appropriate local variables, starting with the value of local variable 1. The individual values must be separated from each other by spaces.

3. If you want to check whether the macro is running:

- Query whether a macro is running on the controller by sending the #8 command.
- Query the name of the macro that is currently running on the controller by sending the RMC? command.

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

**INFORMATION**

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

The axis 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-663 is switched on or rebooted, since they are only written to the volatile memory of the C-663.

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.

#### Example: Implementing multiple calls of a macro via a loop

##### INFORMATION

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

The TESTDION macro checks the status of the digital input lines on the I/O socket. It uses a local variable to identify the digital input line (1 to 4). So that the TESTDION macro does not have to be called separately for each input line, another macro with a loop is recorded.

- Record the LOOPDION macro by sending:

```
MAC BEG loopdion
VAR COUNTER 1
MAC START TESTDION ${COUNTER}
ADD COUNTER ${COUNTER} 1
JRC -2 VAR? COUNTER < 5
MAC END
```

The COUNTER variable is created with the value 1. After this, the TESTDION macro is started for the input line whose identifier is specified via the COUNTER variable. Then the value of the COUNTER is increased by 1. As long as the value of the COUNTER is less than 5, the macro execution pointer subsequently jumps two lines back, so that the TESTDION is now started for the next digital input line.

#### 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-663 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 for operation via a startup macro

#### INFORMATION

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

The STARTCL macro switches the joystick control off for axis 1, activates the motor and starts a reference move to the negative limit switch. As STARTCL is defined as the startup macro, axis 1 is ready for operation immediately after switching on.

- Send:

```
MAC BEG startcl  
JON 1 0  
SVO 1 1  
DEL 1000  
FNL 1  
MAC END  
MAC DEF startcl
```

**INFORMATION**

When this macro is used, the C-663's parameter settings should be adapted to the connected positioner in the nonvolatile memory. Alternatively, the parameter settings can also be configured in the volatile memory via the startup macro. For further information, see "Adapting Settings" (p. 245).

**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.

## 7.7.4 Making Backups and Loading Controller Macros

For example, making backups of controller macros on the PC can be useful before updating the firmware (p. 263).

**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-663 with PIMikroMove**

1. Select the **Controller macros** tab in the PIMikroMove main window.
2. Load macros from the PC to the C-663:
  - 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-663.
  - c) Click **Open**.

For each selected text file (<macro name>.txt), the content is loaded as a macro <macro name> into the C-663.

#### **7.7.5 Macro Example: Synchronization of Two Controllers**

##### **INFORMATION**

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

Action	Command	Result
Connect the digital output line 1 on the I/O socket of the master controller to digital input line 1 on the I/O socket of the slave controller.	- Use a suitable cable. Pin assignment see "I/O" (p. 281).	The digital output signal of the master controller can be used as the trigger for the motion of the axis connected to the slave controller.
Set up the motion on the master controller and on the slave controller.	SVO 1 1 FRF 1 1 VEL 1 0 MOV 1 5.5	For both controllers: The motor is activated and the axis has done a reference move – here to the reference switch. The velocity is set to zero. The axis does not move for now as a result, even though the motion command for the move to absolute position 5.5 has already been sent.

Action	Command	Result
Record the MASTER macro on the master controller.	MAC BEG master DIO 1 1 VEL 1 100 MAC END	The macro has the following tasks: <ul style="list-style-type: none"><li>▪ Switch the digital output line 1 of the master controller to high state to trigger the slave controller</li><li>▪ Set velocity to 100 to start the motion</li></ul>
Record the SLAVE macro on the slave controller.	MAC BEG slave WAC DIO? 1 = 1 VEL 1 100 MAC END	The macro has the following tasks: <ul style="list-style-type: none"><li>▪ Set condition: The macro continues only if digital input line 1 has the high state (i.e., if the master controller outputs the trigger signal).</li><li>▪ Set velocity to 100 to start the motion</li></ul>
Start the SLAVE macro on the slave controller.	MAC START slave	The axis on the slave controller is still not moving because the condition for running the macro further has not yet been met.
Start the MASTER macro on the master controller.	MAC START master	Both axes are moving because their velocity is now each different from zero. The motion occurs synchronously.

## 7.7.6 Macro Example: Stopping Motion by Pushbutton

### INFORMATION

You can connect the C-170.PB pushbutton box from PI to the I/O socket to generate the digital input signals for use in macros. It also displays the state of the digital output lines via LEDs.

### INFORMATION

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

Action	Command	Result
Connect digital input line 1 on the I/O socket to a suitable signal source.	- Pin assignment see "I/O" (p. 281).	The digital input signal can be used for a conditional jump of the macro execution pointer, for example.
Record the HALT macro on the controller.	MAC BEG halt MVR 1 5 JRC 2 DIO? 1 = 1 JRC -1 ONT? 1 = 0 HLT 1 MAC END	The macro has the following tasks: <ul style="list-style-type: none"><li>▪ Start relative motion of axis 1</li><li>▪ Set a condition: If digital input line 1 has the high state (when using the pushbutton box: button 1 is pressed), the macro execution pointer jumps two lines forward. The axis is stopped as a result. Otherwise macro execution is continued with the next line.</li><li>▪ Set a condition: As long as axis 1 has not yet reached the target position, the macro execution pointer jumps back one line. A loop is established as a result.</li></ul>
Start the HALT macro on the controller.	MAC START halt	Axis 1 starts to move. It is stopped by switching digital input line 1 to the high state (e.g., by pushbutton). Regardless of whether the axis has reached the target position or was halted previously, the error code is set to 10 via the HLT command.
If error code 10 interferes: Record alternative HALTVAR macro which uses a variable. See "Variables" (p. 132) details.	MAC BEG haltvar MVR 1 5 JRC 2 DIO? 1 = 1 JRC -1 ONT? 1 = 0 CPY TARGET POS? 1 MOV 1 \${TARGET} VAR TARGET MAC END	The macro has the same tasks as the HALT macro. However, axis 1 is not stopped by pushbutton via the HLT command; instead the result of the POS? 1 query is copied to the TARGET variable. Then this variable is used as the target position for the MOV command. As a result, the axis stays right where it was. To clean up, TARGET is defined as empty with the VAR command which deletes the variable.
Start the HALTVAR macro on the controller.	MAC START haltvar	Axis 1 starts to move. It is stopped by switching digital input line 1 to the high state (e.g., by pushbutton). Error code 10 is not set because no halt or stop command is used.

## 7.7.7 Macro Example: Joystick Control with Storage of Positions

### Task:

Axis 1 is to be controlled with a joystick. Joystick control should only be active when the joystick button is pressed at the same time. By using the buttons of a connected pushbutton box, in addition up to four positions are to be stored in the controller or approached by the axis. The LEDs of the pushbutton box should indicate whether the controller is ready to save the current position and whether it has been saved.

**Approach:**

The STARTUP, MAINLOOP, TESTJOYB, TESTDION, and MVAX2ST macros are recorded on the controller. They use the global variables STORE1, STORE2, STORE3, STORE4, COUNTER, and the local variables 1 and 2.

**INFORMATION**

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

Action	Command	Result
Connect C-170.PB pushbutton box from PI to the <b>I/O</b> socket.	-	Digital input lines 1 to 4 are switched to high state as long as the respective button is pressed. The states of digital output lines 1 to 4 are indicated by the LEDs which are integrated in the buttons.
Connect C-819.20 or C-819.30 joystick to the <b>Joystick</b> socket.	-	For commands, the joystick axis connected is accessible as axis 1 of joystick 1 and the joystick button is accessible as button 1 of joystick 1.
Activate motor for axis 1.	SVO 1 1	The motor must be activated, so that axis 1 can be controlled via a joystick.
Start reference move for axis 1.	FRF 1	The axis starts a reference move – here to the reference switch. After this, absolute axis positions can be commanded.
Specify which axis is to be controlled via the joystick axis.	JAX 1 1 1	Axis 1 is assigned to axis 1 of joystick 1. Joystick control is still not active.
Record the STARTUP macro on the controller.	<pre>MAC BEG startup CPY STORE1 POS? 1 CPY STORE2 POS? 1 CPY STORE3 POS? 1 CPY STORE4 POS? 1 MAC START MAINLOOP MAC END</pre>	<p>The macro has the following tasks:</p> <ul style="list-style-type: none"> <li>▪ Initialize variables for storing the position</li> <li>▪ Start MAINLOOP macro for the main loop</li> </ul>

Action	Command	Result
Record the MAINLOOP macro on the controller.	<pre>MAC BEG mainloop MAC START TESTJOYB VAR COUNTER 1 MAC START TESTDION \${COUNTER} ADD COUNTER \${COUNTER} 1 JRC -2 VAR? COUNTER &lt; 5 MAC START MAINLOOP MAC END</pre>	The macro has the following tasks: <ul style="list-style-type: none"> <li>▪ Start TESTJOYB macro for joystick control</li> <li>▪ Start TESTDION macro successively for all digital inputs (i.e., every pushbutton box button), using a loop</li> <li>▪ Call itself to set up the main loop</li> </ul>
Record the TESTJOYB macro on the controller.	MAC BEG testjoyb	The macro has the following tasks:
	MEX JBS? 1 1 = 0	<ul style="list-style-type: none"> <li>▪ Stop running the macro if joystick button 1 is not pressed</li> </ul>
	JON 1 1	<ul style="list-style-type: none"> <li>▪ Activate joystick 1</li> </ul>
	DIO 0 15	<ul style="list-style-type: none"> <li>▪ Switch all LEDs on the pushbutton box on</li> </ul>
	JRC 6 JBS? 1 1 = 0	<ul style="list-style-type: none"> <li>▪ Jump forward 6 lines (to JON 1 0), if joystick button 1 is no longer pressed</li> </ul>
	DEL 50	<ul style="list-style-type: none"> <li>▪ Wait 50 ms</li> </ul>
	DIO 0 0	<ul style="list-style-type: none"> <li>▪ Switch all LEDs on the pushbutton box off</li> </ul>
	JRC 3 JBS? 1 1 = 0	<ul style="list-style-type: none"> <li>▪ Jump forward 3 lines (to JON 1 0) if joystick button 1 is no longer pressed</li> </ul>
	DEL 50	<ul style="list-style-type: none"> <li>▪ Wait for an additional 50 ms</li> </ul>
	JRC -6 JBS? 1 1 = 1	<ul style="list-style-type: none"> <li>▪ Jump back 6 lines (to DIO 0 15), if joystick button 1 is still pressed</li> </ul>
	JON 1 0	<ul style="list-style-type: none"> <li>▪ Deactivate joystick 1</li> </ul>
	DIO 0 0	<ul style="list-style-type: none"> <li>▪ Switch all LEDs on the pushbutton box off</li> </ul>
	MAC END	
Record the TESTDION macro on the controller.	MAC BEG testdion	The macro has the following tasks:
	MEX VAR? 0 != 1	<ul style="list-style-type: none"> <li>▪ Stop running the macro if the number of local variables specified is not 1 when starting TESTDION</li> </ul>
	MEX DIO? \$1 = 0	<ul style="list-style-type: none"> <li>▪ Stop running the macro if the pushbutton box button specified via local variable 1 is no longer pressed (corresponding input line has the low state)</li> </ul>

Action	Command	Result
	DEL 300	▪ Wait 300 ms
	JRC 3 DIO? \$1 = 1	▪ If the button is still pressed, jump 3 lines forward (to DEL 400)
	MAC START MVAX2ST \$1	▪ Start the MVAX2ST macro because the button was only briefly pressed. The value of the local variable 1 is also used for local variable 1 in MVAX2ST. MVAX2ST moves axis 1 to the position assigned for the button.
	MEX DIO? \$1 = 0	▪ Stop running macro if button is no longer pressed
	DEL 400	▪ Wait 400 ms
	MEX DIO? \$1 = 0	▪ Stop running macro if button is no longer pressed
	DIO \$1 1	▪ Switch the pushbutton box LED on that is associated with the button pressed to indicate storing of the current position
	WAC DIO? \$1 = 0	▪ The macro continues to run only if the button is no longer pressed
	DIO \$1 0	▪ Switch LED off
	CPY STORE\$1 POS? 1	▪ Save the current position of axis 1 in the global variable designated via local variable 1
	MAC END	
Record the MVAX2ST macro on the controller.	MAC BEG MVAX2ST	The macro has the following tasks:
	CPY 2 VAR? STORE\$1	▪ Queries the storage variable designated via local variable 1 and copies its value to local variable 2
	MOV 1 \$2	▪ Start motion of axis 1 to the target position specified via local variable 2
	MAC END	

Action	Command	Result
Start the STARTUP macro on the controller. Alternative: If the variables for storing positions are not to be initialized, start the MAINLOOP macro on the controller instead.	MAC START startup	<p>Joystick control is activated by pressing the joystick button. When joystick control is activated, the pushbutton box LEDs flash rapidly and therefore indicate that the box buttons should not be pressed. After releasing the joystick button, joystick control is deactivated and the LEDs switch themselves off. The pushbutton box can now be used for moving to the saved positions or for saving the current position.</p> <p>The respective button on the pushbutton box is pressed briefly to move the positioner to a stored position.</p> <p>To store the current position of the positioner, a button is pressed on the pushbutton box until the button LED lights up.</p>

### 7.7.8 Macro Example: Joystick Control with Change in Velocity

#### INFORMATION

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

Action	Command	Result
Connect C-819.20 or C-819.30 joystick to the <b>Joystick</b> socket.	-	For commands, the joystick axis connected is accessible as axis 1 of joystick 1 and the joystick button is accessible as button 1 of joystick 1.
Activate motor for axis 1.	SVO 1 1	The motor must be activated, so that axis 1 can be controlled via a joystick.
Start reference move for axis 1.	FRF 1	The axis starts a reference move – here to the reference switch. After this, absolute axis positions can be commanded.
Specify which axis is to be controlled via the joystick axis.	JAX 1 1 1	Axis 1 is assigned to joystick axis 1 of joystick 1. Joystick control is still not activated.
Record the JOYVEL	MAC BEG joyvel	The macro has the following tasks:

Action	Command	Result
macro on the controller.	JON 1 1	▪ Activate joystick 1
	JRC 3 JBS? 1 1 = 1	▪ If joystick button 1 is pressed, jump 3 lines forward (to VEL 1 1).
	VEL 1 0.5	▪ Maximum velocity during joystick control is 0.5.
	JRC -2 JBS? 1 1 = 0	▪ If joystick button 1 is not pressed, jump 2 lines back to set up a loop.
	VEL 1 1	▪ Maximum velocity during joystick control is 1.
	JRC -4 JON? 1 = 1	▪ If joystick 1 is still activated, jump 4 lines back to set up a loop.
	MAC END	
Start the JOYVEL macro on the controller.	MAC START joyvel	Motion with low velocity: Move the joystick control lever. Motion with high velocity: Keep the pushbutton 1 of the joystick pressed and move the control lever.

## 8 GCS Commands

### In this Chapter

Notation .....	129
GCS Syntax for Syntax Version 2.0 .....	129
Target and Sender Address .....	132
Variables .....	132
Command Overview .....	134
Command Descriptions for GCS 2.0 .....	138
Error Codes .....	220

### 8.1 Notation

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

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

### 8.2 GCS Syntax for Syntax Version 2.0

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

Command mnemonic:

CMD ::= character1 character2 character3 [?]

Exceptions:

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

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

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

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

Exception:

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

The argument <AxisID> is used for the logical axes of the controller. Depending on the controller, an axis identifier can consist of up to 16 characters. All alphanumeric characters and the underscore are allowed. Refer to "Commandable Elements (p. 18)" for the identifiers supported by the C-663.

Example 1:

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

Send: MOV[SP]1[SP]10.0[LF]

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

Example 2:

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

Send: MOV[SP]1[SP]17.3[SP]2[SP]2.05[LF]

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

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

Example 3:

All parameters in the volatile memory are to be reset.

Send: RPA[LF]

Example 4:

The position of all axes is to be queried.

Send: POS? **LF**

The response syntax is as follows:

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

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

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

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

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

Query command:

CMD?{:}SP<Arg3>{:}SP<Arg1>{:}SP<Arg2>**LF**

Response to this command:

<Arg3>"=<Val3>**SPLF**

<Arg1>"=<Val1>**SPLF**

<Arg2>"=<Val2>**LF**

Example 5:

Send: TSP?{:}SP2{:}SP1**LF**

Receive: 2=-1158.4405**SPLF**

1=+0000.0000**LF**

### **INFORMATION**

With the C-663 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 only the PC may send command lines to the controllers, its address (0) does not need to be specified. However, both target and sender address are part of any controller response. Multiline responses contain the target and sender address in the first line

Example:

In a terminal, e.g., PITerminal, the identification string of a controller with address 2 (here: a C-863.11) is queried using the \*IDN? command.

Send: 2 0 \*IDN?

or

Send: 2 \*IDN?

The response in either case is:

```
0 2 (c)2011 Physik Instrumente(PI) Karlsruhe, C-863.11,0,1.2.0.0
```

Exception:

The target address can be omitted if the target controller has the address 1, even if this controller is part of a daisy chain. If the target address is not specified when addressing a controller, the target and sender addresses will also not be specified in the response from the controller.

Example:

Send: \*IDN?

The controller with address 1 (here: a C-863.11) responds with:

```
(c)2011 Physik Instrumente(PI) Karlsruhe, C-863.11,0,1.2.0.0
```

Send: 1 \*IDN?

The same controller responds with:

```
0 1 (c)2011 Physik Instrumente(PI) Karlsruhe, C-863.11,0,1.2.0.0
```

See "Adapting DIP Switch Settings" (p. 60) on how to set the controller address. The controller address can be in the range of 1 to 16, address 1 is default. The PC always has the address 0. With the broadcast address 255, all controllers in a daisy chain can be addressed at the same time, but no responses are sent to the PC.

## 8.4 Variables

For more flexible programming, the C-663 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 specified 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> indicate the values for the local variables 1 and 2 used in the macro. <STRING1> and <STRING2> can be specified directly or via the values of variables. <uint> defines the number of times the macro is to be run. See the `MAC` command (p. 184) description for more information.

- The local variable 0 is read-only. Its value gives the number of arguments (i.e., values of local variables) set when starting the macro.
- Inside a macro, the values of local variables can be modified using `ADD` (p. 142), `CPY` (p. 147) or `VAR` (p. 215), and can be deleted with `VAR` (except for the 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 `ADD`, `CPY` or `VAR`. 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. 138)
#5		Request Motion Status (p. 139)
#7		Request Controller Ready Status (p. 140)
#8		Query If Macro Is Running (p. 140)
#24		Stop All Axes (p. 140)
*IDN?		Get Device Identification (p. 141)
ACC	{<AxisID> <Acceleration>}	Set Closed-Loop Acceleration (p. 141)
ACC?	[{<AxisID>}]	Get Closed-Loop Acceleration (p. 142)
ADD	<Variable> <FLOAT1> <FLOAT2>	Add and Save To Variable (p. 142)
BRA	BRA {<AxisID> <BrakeState>}	Set Brake Activation State (p. 145)
BRA?	BRA? [{<AxisID>}]	Get Brake Activation State (p. 145)
CCL	<Level> [<PSWD>]	Set Command Level (p. 146)
CCL?		Get Command Level (p. 147)
CPY	<Variable> <CMD?>	Copy Into Variable (p. 147)
CST?	[{<AxisID>}]	Get Assignment Of Stages To Axes (p. 148)
CSV?		Get Current Syntax Version (p. 148)
CTO	{<TrigOutID> <CTOPam> <Value>}	Set Configuration Of Trigger Output (p. 149)
CTO?	[{<TrigOutID> <CTOPam>}]	Get Configuration Of Trigger Output (p. 153)

<b>Com- mand</b>	<b>Arguments</b>	<b>Description</b>
DEC	{<AxisID> <Deceleration>}	Set Closed-Loop Deceleration (p. 153)
DEC?	[{<AxisID>}]	Get Closed-Loop Deceleration (p. 154)
DEL	<uint>	Delay The Command Interpreter (p. 154)
DFH	[{<AxisID>}]	Define Home Position (p. 155)
DFH?	[{<AxisID>}]	Get Home Position Definition (p. 156)
DIO	{<DIOID> <OutputOn>}	Set Digital Output Lines (p. 157)
DIO?	[{<DIOID>}]	Get Digital Input Lines (p. 158)
DRC	{<RecTableID> <Source> <RecOption>}	Set Data Recorder Configuration (p. 158)
DRC?	[{<RecTableID>}]	Get Data Recorder Configuration (p. 160)
DRL?	[{<RecTableID>}]	Get Number Of Recorded Points (p. 160)
DRR?	[<StartPoint> <NumberOfPoints> [{<RecTableID>}]]	Get Recorded Data Values (p. 161)
DRT	{<RecTableID> <TriggerSource> <Value>}	Set Data Recorder Trigger Source (p. 162)
DRT?	[{<RecTableID>}]	Get Data Recorder Trigger Source (p. 163)
ERR?		Get Error Number (p. 164)
FED	{<AxisID> <EdgeID> <Param>}	Find Edge (p. 165)
FNL	[{<AxisID>}]	Fast Reference Move To Negative Limit (p. 166)
FPL	[{<AxisID>}]	Fast Reference Move To Positive Limit (p. 167)
FRF	[{<AxisID>}]	Fast Reference Move To Reference Switch (p. 168)
FRF?	[{<AxisID>}]	Get Referencing Result (p. 169)
GOH	[{<AxisID>}]	Go To Home Position (p. 170)
HDR?		Get All Data Recorder Options (p. 170)
HPA?		Get List of Available Commands (p. 171)
HLT	[{<AxisID>}]	Halt Motion Smoothly (p. 172)
HPA?		Get List Of Available Parameters (p. 172)
HPV?	HPV?	Get List Of Possible Parameter Values (p. 174)
JAS?	[{<JoystickID> <JoystickAxis>}]	Query Joystick Axis Status (p. 175)
JAX	<JoystickID> <JoystickAxis> <AxisID>	Set Axis Controlled By Joystick (p. 176)

Com-mand	Arguments	Description
JAX?	[{<JoystickID> <JoystickAxis>}]	Get Axis Controlled By Joystick (p. 176)
JBS?	[{<JoystickID> <JoystickButton>}]]	Query Joystick Button Status (p. 177)
JDT	{<JoystickID> <JoystickAxis> <uint>}	Set Joystick Default Lookup Table (p. 177)
JLT	[<JoystickID> <JoystickAxis> <Addr> <floatn>]	Fill Joystick Lookup Table (p. 178)
JLT?	[<StartPoint> <NumberOfPoints> [{<JoystickID> <JoystickAxis>}]]]	Get Joystick Lookup Table Values (p. 179)
JON	{<JoystickID> <uint>}	Set Joystick Activation Status (p. 181)
JON?	[{<JoystickID>}]	Get Joystick Activation Status (p. 181)
JRC	<Jump> <CMD?> <OP> <Value>	Jump Relatively Depending On Condition (p. 182)
LIM?	[{<AxisID>}]	Indicate Limit Switches (p. 183)
MAC	<keyword> [{<parameter>}] BEG <macro> DEF <macro> DEF? DEL <macro> END ERR? NSTART <macro> <uint> [<String1> [<String2>]] START <macro> [<String1> [<String2>]]	Call Macro Function (p. 184)
MAC?	[<macro name>]	List Macros (p. 186)
MAN?	<CMD>	Get Help String For Command (p. 187)
MAT	<Variable> <=> <FLOAT1> <OP> <FLOAT2>	Calculate And Save To Variable (p. 188)
MEX	<CMD?> <OP> <Value>	Stop Macro Execution Due To Condition (p. 189)
MOV	{<AxisID> <Position>}	Set Target Position (p. 191)
MOV?	[{<AxisID>}]	Get Target Position (p. 191)
MVR	{<AxisID> <Distance>}	Set Target Relative To Current Position (p. 192)
ONT?	[{<AxisID>}]	Get On-Target State (p. 193)
POS	{<AxisID> <Position>}	Set Real Position (p. 194)
POS?	[{<AxisID>}]	Get Real Position (p. 194)
RBT		Reboot System (p. 195)

<b>Com- mand</b>	<b>Arguments</b>	<b>Description</b>
RMC?		List Running Macros (p. 195)
RON	{<AxisID> <ReferenceOn>}	Set Reference Mode (p. 196)
RON?	[{<AxisID>}]	Get Reference Mode (p. 196)
RPA	[{<ItemID> <PamID>}]	Reset Volatile Memory Parameters (p. 197)
RTR	<RecordTableRate>	Set Record Table Rate (p. 198)
RTR?		Get Record Table Rate (p. 198)
SAI	{<AxisID> <NewIdentifier>}	Set Current Axis Identifiers (p. 199)
SAI?	[ALL]	Get List Of Current Axis Identifiers (p. 199)
SEP	<Pswd> {<ItemID> <PamID> <PamValue>}	Set Nonvolatile Memory Parameters (p. 200)
SEP?	[{<ItemID> <PamID>}]	Get Nonvolatile Memory Parameters (p. 201)
SPA	{<ItemID> <PamID> <PamValue>}	Set Volatile Memory Parameters (p. 202)
SPA?	[{<ItemID> <PamID>}]	Get Volatile Memory Parameters (p. 204)
SRG?	{<AxisID> <RegisterID>}	Query Status Register Value (p. 204)
STE	<AxisID> <Amplitude>	Start Step And Response Measurement (p. 206)
STP		Stop All Axes (p. 206)
SVO	{<AxisID> <MotorState>}	Set Motor State (p. 207)
SVO?	[{AxisID}]	Get Motor State (p. 208)
TAC?		Tell Number Of Analog Input Lines (p. 208)
TAV?	[{<AnalogInputID>}]	Get Analog Input Voltage (p. 209)
TCV?	[{AxisID}]	Get Commanded Closed-Loop Velocity (p. 209)
TIO?		Tell Number Of Digital I/O Lines (p. 209)
TMN?	[{<AxisID>}]	Get Minimum Commandable Position (p. 210)
TMX?	[{<AxisID>}]	Get Maximum Commandable Position (p. 211)
TNR?		Get Number Of Record Tables (p. 211)
TRO	{<TrigOutID> <TrigMode>}	Set Trigger Output State (p. 211)

Com-mand	Arguments	Description
TRO?	[{<TrigOutID>}]	Get Trigger Output State (p. 212)
TRS?	[{<AxisID>}]	Indicate Reference Switch (p. 213)
TSP	{<InputSignalID> <Position>}	Set Input Signal Position Value (p. 213)
TSP?	[{<InputSignalID>}]	Get Input Signal Position Value (p. 214)
TVI?		Tell Valid Character Set For Axis Identifiers (p. 215)
VAR	<Variable> <String>	Set Variable Value (p. 215)
VAR?	[{<Variable>}]	Get Variable Value (p. 216)
VEL	{<AxisID> <Velocity>}	Set Closed-Loop Velocity (p. 217)
VEL?	[{<AxisID>}]	Get Closed-Loop Velocity (p. 217)
VER?		Get Versions Of Firmware And Drivers (p. 218)
WAC	<CMD?> <OP> <Value>	Wait For Condition (p. 218)
WPA	<Pswd> [{<ItemID> <PamID>}]	Save Parameters To Non-Volatile Memory (p. 219)

## 8.6 Command Descriptions for GCS 2.0

### #4 (Request Status Register)

Description: Requests system status information.

Format: #4

Arguments: None

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

Notes: This command is identical in function to SRG? (p. 204) except that only one character is sent via the interface. Therefore, #4 can also be used when the controller is doing time-consuming tasks.

For the C-663, 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	Is referencing	In motion	Motor on	-	-	-	Error flag

Bit	7	6	5	4	3	2	1	0
Description	Digital input line 4	Digital input line 3	Digital input line 2	Digital input line 1	-	Positive limit switch	Reference switch	Negative limit switch

Example:

Send: #4

Receive: 0x9005

Note: The response is in hexadecimal format. It means that the axis is on target (on-target state =true), the motor is activated, no error has occurred, the states of digital input lines 1 to 4 are low, and the positioner is on the positive side of the reference switch (limit switches are not active; note that the logic of the signals is inverted in this example).

## #5 (Request Motion Status)

Description: Requests motion status of the axes.

Format: #5

Arguments: None

Response: The response <uint> is bit-encoded 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  
...

Examples: 0 indicates motion of all axes complete  
3 indicates that the first and second axis are in motion

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

Description: Asks controller for ready status (tests if controller is ready to run a new command).

Note: Use #5 (p. 139) instead of #7 to verify if motion has ended.

Format: #7

Arguments: None

Response: B1h (ASCII character 177 = "±" in Windows) if controller is ready

B0h (ASCII character 176 = "°" in Windows) if controller is not ready  
(e.g., doing a reference move)

Troubleshooting: The response characters may appear differently in non-Western character sets or other operating systems.

**#8 (Query if Macro Is Running)**

Description: Tests if a macro is running on the controller.

Format: #8

Arguments: None

Response: <uint>=0 no macro is running  
<uint>=1 a macro is currently running

**#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. 206), 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. 191), MVR (p. 192), GOH (p. 170), STE (p. 206)), commands for referencing (FNL (p. 166), FPL (p. 167), FRF (p. 168)), and macros (MAC (p. 184)). Also stops macro running.

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

HLT (p. 172) in contrast to #24 stops motion with specified deceleration regarding 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: For C-663, \*IDN? responds something like:

```
(c) 2020 Physik Instrumente (PI) GmbH & Co. KG, C-663.12,116026229,1.2.0.0
```

#### **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/ $s^2$ .

Response: None

Troubleshooting: Illegal axis identifier

Notes: The lowest possible value for <Acceleration> is 0.

ACC changes the value of the parameter ***Closed-Loop Acceleration (Phys. Unit/s<sup>2</sup>)*** (0xB) in the volatile memory of the C-663. The parameter value can be stored as default with WPA (p. 219), refer to "Adapting Settings" "for details" (p. 245).

The maximum value which can be set with the ACC command is specified by the ***Maximum Closed-Loop Acceleration (Phys. Unit/s<sup>2</sup>)*** parameter (0x4A).

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

Description: Queries the acceleration value set with ACC (p. 141).

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. 132).

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:** Local variables can be set using ADD in macros only.

**Example 1:** Value \$B is added to value \$A, and the result is saved to variable C:

```
ADD C $A $B
```

**Example 2:** The name of the variable where the result is to be copied is specified 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
```

**Example 3:** The macros below can be used to create a "flashing light"

with LEDs that are connected to the digital output lines of the controller. \$1 and \$2 are values of local variables and must be specified as arguments of MAC START or MAC NSTART command when starting the macros (see below).

DIO 0 <bitmask>: Sets the output channels according to <bitmask>. For example, "DIO 0 5" activates channels 1 and 3 and deactivates all other channels (5 is 0000 0101 in binary notation).

To implement the "flashing light", do the following steps:

1. Write the "STEPS" macro:

```
MAC BEG STEPS
DIO 0 $1
ADD 1 $1 1
DEL $2
JRC -3 VAR? 1 <= 15
ADD 1 $1 -1
DIO 0 $1
DEL $2
JRC -3 VAR? 1 > 0
MAC END
```

2. Write the "TEST" macro:

```
MAC BEG TEST
MAC START STEPS 0 $1
ADD 1 $1 10
JRC -2 VAR? 1 < 110
VAR 1 10
ADD 2 $2 -1
JRC -5 VAR? 2 > 0
MAC END
```

3. Start the TEST macro with arguments that define the variable values \$1 and \$2:

```
MAC START Test 10 50
```

Meaning of the variables here:

\$1: Delay in ms between each step in the STEPS macro. The value is incremented by 10 by the TEST macro until it reaches 110.

\$2: Number of repetitions of the whole "flashing light" procedure.

**BRA (Set Brake Activation State)**

Description: Activates/deactivates brake for specified axes.

Format: BRA {<AxisID> <BrakeState>}

Arguments: <AxisID> is one axis of the controller

<BrakeState> can have the following values:

0 = Brake deactivated

1 = Brake activated

Response: None

Troubleshooting: Illegal axis identifier

Notes: The brake can only be used if parameter 0x1A (**Has Brake?**) has the value 1 ("yes").

If parameter 0x1A (**Has Brake?**) has the value 1 ("yes"), the following applies:

- The brake can be activated or deactivated by BRA only if the motor is deactivated. Secure the positioner against moving unintentionally before you deactivate the brake before you deactivate the brake with BRA!
- Activating the motor with SVO (p. 207) influences the brake's activation state:
  - Activating the motor deactivates the brake.
  - Deactivating the motor activates the brake.
- If motion error occurs the motor is deactivated and the brake is activated.

If the integrated brake driver of the C-663 is to be used, parameter 0x3094 (**Internal Brake**) must also have the value 1. For more information, see the descriptions in "Parameter Overview" (p. 254).

**BRA? (Get Brake Activation State)**

Description: Gets brake activation state of specified axes.

If all arguments are left out, gets state of all axes.

Format: BRA? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller

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

where

<BrakeState> is the current brake activation state of the axis:

- 0 = Brake deactivated
- 1 = Brake activated

Troubleshooting: Illegal axis identifier

### **CCL (Set Command Level)**

Description: Changes the active "command level" and therefore determines the availability of commands and write access to system parameters.

Format: CCL <Level> [<PSWD>]

Arguments: <Level> is a command level of the controller

<PSWD> is the password required for changing to the appropriate command level

The following command levels and passwords apply:

Level = 0 is the default setting, all commands provided for "normal" users are available, read access to all parameters, no password required.

Level = 1 adds additional commands and write access to level-1 parameters (commands and parameters from level 0 are included). The password required is "advanced".

Level > 1 is provided for PI service personnel only. Users cannot change to a level > 1. Contact the customer service department if you have problems with parameters for command level 2 or higher (p. 273).

Response: None

Troubleshooting: Invalid password

Notes: With the C-663, the command levels only determine the write permission for the parameters. The availability of the commands of the C-663 is independent of the active command level.

HPA? (p. 172) 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. 245).

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: <Level> should be 0 or 1.

<Level> = 0 is the default setting, write access is specified for level 0 parameters, read access is specified for all parameters

<Level> = 1 allows write access for level 1 parameters (parameters from level 0 are included).

### **CPY (Copy Into Variable)**

Description: Copies a command response to a variable (p. 132).

The variable is present in volatile memory (RAM) only.

Format: CPY <Variable> <CMD?>

Arguments: <Variable> is the name of the variable to which the command response is to be copied.

<CMD?> is one query command in its usual notation. The response has to be a single value and not more.

Response: None

Notes: Local variables can be set using CPY in macros only.

**Example 1:** Using the following macro, it is possible to connect through the digital input and output lines of the controller. 1 is a local variable whose value must be specified as argument of the MAC START or MAC NSTART command when starting the macro.

Write the "connect" macro:

```
MAC BEG connect
CPY 1 DIO? 0
DIO 0 $1
MAC START CONNECT
MAC END
```

**Example 2:** It is possible to copy the value of one variable (e.g., SOURCE) to another variable (e.g., TARGET):

```
CPY TARGET VAR? SOURCE
```

### CST? (Get Assignment Of Stages To Axes)

Description: Returns the name of the connected positioner type for the queried axis.

Format: CST? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller

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

where

<string> is the name of the positioner type assigned to the axis.

Notes: The positioner name is read from the **Stage Name** parameter (ID 0x3C) whose factory default value is "DEFAULT\_STAGE". You can set the parameter value to the name of your positioner with SPA (p. 202) or SEP (p. 200). You can find details in the parameter overview (p. 254).

### 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

Notes: 1.0 (for GCS 1.0) or 2.0 (for GCS 2.0) are possible responses.

### **CTO (Set Configuration Of Trigger Output)**

Description: Configures the trigger output conditions for the specified digital output line.

Format: CTO {<TrigOutID> <CTOPam> <Value>}

Arguments: <TrigOutID> is one digital output line of the controller, see below for details.

<CTOPam> is the CTO parameter ID in decimal format, see below for the available IDs.

<Value> is the value that the CTO parameter is set to, see below.

Response: None

Notes: The trigger output conditions will become active when enabled with TRO (p. 211). Do not use DIO (p. 157) on digital output lines where the trigger output is activated by TRO.

The CTO settings are lost when you power down or reboot the C-663. They can be easily maintained by saving them in a macro.

Output lines and trigger conditions available:	<p>&lt;TrigOutID&gt; corresponds to digital output lines 1 to 4, IDs = 1 to 4; see "I/O" (p. 281).</p> <p>&lt;CTOPam&gt; parameter IDs available for C-663:</p> <ul style="list-style-type: none"> <li>1 = TriggerStep</li> <li>2 = Axis</li> <li>3 = TriggerMode</li> <li>7 = Polarity</li> <li>8 = StartThreshold</li> <li>9 = StopThreshold</li> <li>10 = TriggerPosition</li> </ul> <p>&lt;Value&gt; available for the appropriate &lt;CTOPam&gt; ID:</p> <p>for TriggerStep: Distance</p> <p>for Axis: The identifier of the axis to be connected to the digital output line. Irrelevant for the MotionError trigger mode.</p> <p>for TriggerMode (default value is 0):</p> <ul style="list-style-type: none"> <li>▪ 0 = PositionDistance; a trigger pulse is written whenever the axis has covered the TriggerStep distance (&lt;CTOPam&gt; ID 1). Optionally, values for StartThreshold and StopThreshold (&lt;CTOPam&gt; IDs 8 and 9) can be defined to activate the trigger output for a limited position range and a certain direction of motion only (negative or positive; Note: If the motion direction is reversed before the axis position has reached the stop threshold, trigger pulses will continue to be generated). When StartThreshold and StopThreshold are set to the same value, they will not be used.</li> <li>▪ 2 = OnTarget; the on-target state of the selected axis is transferred to the selected digital output line (this state can also be read with the ONT? command).</li> <li>▪ 5 = MotionError; the selected digital output line becomes active when a motion error occurs. The line will stay active until the error code is reset to 0 (by a query).</li> <li>▪ 6 = InMotion; the selected digital output line is active as long as the selected axis is in motion (the motion state can also be read with commands, e.g. SRG? or #5).</li> </ul>
--	--

- 7 = Position+Offset;  
the first trigger pulse is written when the axis has reached the position specified by TriggerPosition (<CTOPam> ID 10). The next trigger pulses are written each time the axis position equals the sum of the last valid trigger position and the distance specified by TriggerStep (<CTOPam> ID 1). Trigger output ends when the axis position exceeds the value specified by StopThreshold (<CTOPam> ID 9). The sign of the TriggerStep value determines the direction of motion in which trigger pulses are to be output. Trigger processing is done by the DSP of the C-663.
- 8 = SinglePosition;  
the selected digital output line is active when the axis position has reached or exceeded the position specified by TriggerPosition (<CTOPam> ID 10).

for Polarity (default value is 1): sets the signal polarity for the digital output line  
0 = Active Low  
1 = Active High

for StartThreshold/StopThreshold: position value;  
if used for the PositionDistance trigger mode, both thresholds must be set in order to determine the position range and the direction of motion for trigger output;  
StopThreshold is used as the stop condition for Position+Offset trigger mode

for TriggerPosition: position value;  
if used in the Position+Offset trigger mode, the first trigger pulse is output at this position;  
if used in the SinglePosition trigger mode, the output line is active when this position is reached or exceeded

For application examples and further details see "Digital Output Signals" (p. 89) and the lines below.

**Example 1:** A pulse is to be generated on digital output line 1 (ID 1) whenever axis 1 has covered a distance of 0.05 µm. The following parameters must be set:

```
TrigOutID = 1
Axis = 1
TriggerMode = 0
TriggerStep = 0.05
Send: CTO 1 2 1
Send: CTO 1 3 0
Send: CTO 1 1 0.00005
```

**Example 2:**

In this example, digital output line 1 is to be set from low to high when axis A starts to move. The following parameters must be set:

TrigOutID = 1

Axis = A (axis identifier was changed with SAI)

TriggerMode = 6

Polarity = Active High

So you have to send:

CTO 1 2 A

CTO 1 3 6

CTO 1 7 1

**Example 3:**

L-509.1xxxxx (travel range: 26 mm) is connected to axis 1. The reference position of the L-509.1xxxxx is 13 mm.

Starting from its reference position, the axis is to be moved alternating forwards and backwards; trigger pulses are to be output with the Position+Offset trigger mode for both directions of motion in a range of 1 mm. For that purpose, two macros are written to the controller. Macro TRIGREF initializes the controller and could also be defined as startup macro, while macro TRIGGER starts motion and therefore trigger output. Write the macros as shown below. Further details about macros see "Working with Macros" (p. 112).

Make sure that the velocity for the axis matches the CTO setting for the distance. Recommended value:

maximum velocity = distance \* 20 kHz / 2

where 20 kHz is the frequency of the C-663 servo cycle.

A trigger signal frequency of 1 kHz results at a velocity of 20 mm/s.

➤ Record a macro named TRIGREF with the following contents:

CTO 1 3 7

SVO 1 1

FRF

TRO 1 1

MAC START TRIGGER

➤ Record a macro named TRIGGER with the following contents:

CTO 1 1 0.02

CTO 1 9 15

CTO 1 10 14

```

DEL 1000
DRT 0 2 0
MOV 1 15.01
WAC POS? 1 > 14.8
MEX CTO? 1 10 < 13.9
CTO 1 1 -0.02
CTO 1 9 14
CTO 1 10 15
DEL 1000
MOV 1 13.99
WAC POS? 1 < 14
MEX CTO? 1 10 > 14.1
MAC START TRIGGER

```

### **CTO? (Get Configuration Of Trigger Output)**

Description: Queries the values set for specified trigger output lines and parameters.

Format: CTO? [{<TrigOutID>} <CTOPam>]}

Arguments: <TrigOutID>: is a digital output line of the controller; see CTO.

<CTOPam>: parameter ID; see CTO.

If all arguments are left out, the response contains the values for all parameters and all output lines.

Response: {<TrigOutID>} <CTOPam>"=<Value> LF}

For <Value> see CTO.

### **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/ $s^2$ .

Response: None

Troubleshooting: Illegal axis identifier

Notes: The lowest possible value for <Deceleration> is 0.

DEC changes the value of the parameter ***Closed Loop Deceleration (Phys. Unit/s<sup>2</sup>)*** (0xC) in the volatile memory of the C-663. The parameter value can be stored as default with WPA (p. 219), see "Adapting Settings" (p. 245) for details.

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

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

Description: Queries the deceleration value set with DEC (p. 153).

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/ $s^2$ .

### 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. 184) and in the "Controller Macros" (p. 110) 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. 194) (Get the current position)
- TMN? (p. 210) (Get the minimum commandable position)
- TMX? (p. 211) (Get the maximum commandable position)

The offset is reset to zero in the following cases:

- When switching on and rebooting the C-663
- With referencing

The following values are **not** changed with DFH:

- Output value for TSP? (p. 214) (Get the current sensor position)
- Parameters for the definition of travel range and soft limits (p. 34)

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: Queries the position value that is currently used as the offset for the specified axis to move the zero position.

If no arguments are specified, queries the position value of all axes.

Format: DFH? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller

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

where

<PositionOffset> 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.

Troubleshooting: Illegal axis identifier

Notes: The offset is reset to zero in the following cases:

- When switching on and rebooting the C-663
- When referencing

See DFH for an example.

### **DIO (Set Digital Output Line)**

Description: Switches the specified digital output line(s) to specified state(s).

Use TIO? (p. 209) to get the number of installed digital I/O lines.

Format: DIO {<DIOID> <OutputOn>}

Arguments: <DIOID> is one digital output line of the controller, see below for details.

<OutputOn> is the state of the digital output line, see below for details.

Response: none

Notes: You can use the DIO command to activate/deactivate the Output 1 to Output 4 lines on the I/O socket (p. 281). The C-663 allows you to either set a single line per DIO command, or all lines at once.

The <DIOID> identifiers to use for the lines are 1 to 4. With the identifier 0, all lines are set according to a bit pattern specified by <OutputOn>.

If <OutputOn>=1 the line is set to HIGH/ON, if <OutputOn>=0 it is set to LOW/OFF.

Do not use DIO on output lines for which the trigger output is activated with TRO (p. 211).

**DIO? (Get Digital Input Lines)**

Description: Gets the states of the specified digital input lines.

Use TIO? (p. 209) to get the number of available digital I/O lines.

Format: DIO? [{<DIOID>}]

Arguments: <DIOID> is the identifier of the digital input line, see below for details.

Response: {<DIOID>=<InputOn> LF}

where

<InputOn> specifies the state of the digital input line, see below for details.

Notes: You can use the DIO? command to read digital input lines 1 to 4 on the **I/O** socket directly (p. 281).

The <DIOID> identifiers to use for the lines are 1 to 4. If the identifier is left out or 0, all lines are queried.

If <InputOn>=0, the digital input is LOW/OFF; if <InputOn>=1, the digital input is HIGH/ON. If <DIOID> is 0, <InputOn> is a bit pattern which gives the states of all lines in hexadecimal format.

**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-663 has 4 data recorder tables with 1024 points per table.

With HDR? (p. 170), 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. 211).

Refer to "Data Recorder" (p. 87) for further information.

**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 axis
- 3=Position error of axis
- 70=Commanded velocity of axis
- 71=Commanded acceleration of axis
- 73=Motor output of axis (dimensionless control value)
- 74=K<sub>p</sub> of axis
- 75=K<sub>i</sub> of axis
- 76=K<sub>d</sub> of axis
- 77=K<sub>v</sub> of axis
- 80=Signal status register of axis
- 90=active parameter set (only relevant if the controller supports several groups of servo control parameters)

Data source is the analog input:

- 81=Analog input (channel = 1 - 4)

Data source is a motor output (see pin assignment (p. 279)):

- 100=Current Phase A [mA]
- 101=Current Phase B [mA]

Note: The analog inputs for the record option 81 can be the Input 1 to Input 4 lines of the I/O socket (p. 281). Use the identifiers 1 to 4 for these data sources.

Data source identifiers 5 and 6 refer to the inputs for the joystick's axis and button:

5 = Axis 1 of the joystick

6 = Button 1 of the joystick

**DRC? (Get Data Recorder Configuration)**

- Description: Queries the settings for the data to be recorded.
- Format: DRC? [{<RecTableID>}]
- Arguments: <RecTableID>: is a data recorder table of the controller; if this entry is not specified, the response will contain the settings for all tables.
- Response: The current DRC settings:  
 $\{<\text{RecTableID}> = "<\text{Source}> <\text{RecOption}> \text{LF}\}$   
 where  
<Source>: is the data source, for example, an axis or a channel of the controller. The source type depends on the record option.  
<RecOption>: is the type of data to be recorded (record option).  
 The available record options can be queried with HDR? (p. 170).

**DRL? (Get Number of Recorded Points)**

- Description: Reads the number of points comprised by the last recording.
- Format: DRL? [{<RecTableID>}]
- Arguments: <RecTableID> is one data recorder table of the controller
- Response: {<RecTableID> = "<uint> \text{LF}"  
 where  
<uint> specifies the number of points recorded with the last recording
- Notes: The number of points is reset to zero for a data recorder table when changing its configuration with DRC (p. 158).

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

Description: Gets the last recorded data.

Reading can take some time depending on the number of points to be read!

It is possible to read the data while recording is still in progress.

Format: DRR? [<StartPoint> <NumberOfPoints> [<RecTableID>]]

Arguments: <StartPoint> is the first point to be read from the data recorder table; it starts with index 1.

<NumberOfPoints> is the number of points to be read per table.

<RecTableID> is one data recorder table of the controller.

Response: For the recorded data in GCS array format, refer to the separate manual for the GCS array, SM146E, and the example below.

Notes: If <RecTableID> is not specified, the data is read from all tables with a record option not equal to zero.

With HDR? (p. 170), you will obtain a list of all available recording and triggering options as well as additional information on data recording.

Refer to the description of the DRC command (p. 158) as well as "Data Recorder" (p. 87) for further information.

Example:

```
rtr?
10
drr? 1 20
# REM C-663
#
# VERSION = 1
# TYPE = 1
# SEPARATOR = 32
# DIM = 2
# SAMPLE_TIME = 0.000500
# 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
5.00000 0.00000
5.00000 0.00000
4.99998 0.00002
5.00000 0.00000
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 specified trigger source is set for all data recorder tables that have a record option that is not zero.

Irrespective of the trigger option set, data recording is always triggered when step response measuring is done with STE (p. 206).

With HDR? (p. 170), 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. 158) as well as "Data Recorder" (p. 87).

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. 192), MOV (p. 191); <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.

### **DRT? (Get Data Recorder Trigger Source)**

Description: Gets the trigger source for the data recorder tables.

Format: DRT? [{<RecTableID>}]

Arguments: <RecTableID> is one data recorder table of the controller.

Response: {<RecTableID>}"=<TriggerSource> <Value> LF}

where

<TriggerSource> is the identifier of the trigger source.

<Value> depends on the trigger source.

Further information can be found in the description of the DRT command (p. 162).

**Notes:** Because all data record tables of the C-663 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:** Get error code <int> of the last occurred error and reset the error to 0.

Only the last error is buffered. You should therefore call ERR? after each command.

The error codes and their descriptions are listed in "Error Codes" (p. 220).

**Format:** ERR?

**Arguments:** None

**Response:** The error code of the last error that occurred (integer).

**Troubleshooting:** Communication breakdown

**Notes:** In the case of simultaneous access to the controller by several instances, the error code is only returned to the first instance that sent the ERR? command. Because the error is reset to 0 by the query, the error is not visible for any further querying instance.

- If possible, access the controller with one instance only.
- If incorrect system behavior does not cause the controller to send an error code, check whether the error code is queried regularly in the background by a macro, script or the PC software (e.g., PIMikroMove).

If the cause of an error continues, the corresponding error code is immediately set again after a query with ERR?.

**FED (Find Edge)**

Description: Moves the specified axis to a specified signal edge.

FED does not set a certain position value at the selected edge (in contrast to the FNL (p. 166), FPL (p. 167), and FRF (p. 168) commands for referencing), i.e., the axis is not "referenced" after using FED.

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

Format: FED {<AxisID> <EdgeID> <Param>}

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

<EdgeID> is the type of edge the axis has to move to. See 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. 208) responds with the value 0.

Notes: The motor must be activated with SVO (p. 207) for the commanded axis before using this command.

The C-663 firmware detects the presence or absence of reference switches and limit switches using parameters (ID 0x14 for reference switch; ID 0x32 for limit switches). According to the values of those parameters, the C-663 activates or deactivates FED motion to the corresponding signal edges. Adapt the parameter values to your hardware using SPA (p. 202) or SEP (p. 200). Refer to "Parameter Overview" (p. 254) for further information.

You can use the digital input lines instead of the switches as source of the switch signals for FED. Refer to "Digital Input Signals" (p. 96) for further information.

FED can be used to measure the physical travel range of new mechanics and therefore to identify the values for the corresponding parameters: The distance from the negative to the positive limit switch, the gap between the negative limit switch and the reference switch (parameter ID 0x17), and the gap between reference switch and positive limit

switch (parameter ID 0x2F). Refer to "Travel Range and Soft Limits" (p. 34) for further information.

Motion can be stopped by #24 (p. 140), STP (p. 206), and HLT (p. 172).

Motion commands such as FED are not permitted when the joystick is active for the axis. Refer to "Joystick Control" (p. 102) for further information.

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: The motor must be activated with SVO (p. 207) for the commanded axis before using this command.  
If the reference move was successful, absolute motion will then be possible.

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.

You can use a digital input instead of the negative limit switch as source of the negative limit switch signal for FNL. Refer to "Digital Input Signals" (p. 96) for further information.

The motion can be stopped by #24 (p. 140), STP (p. 206) and HLT (p. 172).

Use FRF? (p. 169) 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.

Refer to "Referencing" (p. 38) and "Travel Range and Soft Limits" (p. 34) for further information.

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

Description: Starts a reference move.

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.

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

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: The motor must be activated with SVO (p. 207) for the commanded axis before using this command.

If the reference move was successful, absolute motion will then be possible.

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.

You can use a digital input instead of the positive limit switch as source of the positive limit switch signal for FPL. For further information, see "Digital Input Signals" (p. 96).

Motion can be stopped by #24 (p. 140), STP (p. 206) and HLT (p. 172).

Use FRF? (p. 169) 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. 38) and "Travel Range and Soft Limits" (p. 34).

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

Description: Starts a reference move.

Moves the specified axis to the reference switch and sets the current position to a defined value. See below for details.

If multiple axes are specified in the command, they are started simultaneously.

Format: FRF [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller, if left out, all axes are affected.

Response: None

Troubleshooting: Illegal axis identifier

Notes: The motor must be activated with SVO (p. 207) for the commanded axis before using this command.  
If the reference move was successful, absolute motion will then be possible.

The value of the parameter 0x16 is set as the current position when the axis is at the reference switch.

You can use a digital input instead of the reference switch as source of the reference signal for the FRF command. For further information, see "Digital Input Signals" (p. 96).

The motion can be stopped by #24 (p. 140), STP (p. 206) and HLT (p. 172).

Use FRF? (p. 169) to check whether the reference move was successful.

Use FNL (p. 166) or FPL (p. 167) instead of FRF (p. 168) to do a reference move for an axis that has no reference switch but limit switches.

For best repeatability, referencing must always be done in the same way. The FRF command always approaches the reference switch from the same side, no matter where the axis is when the command is called.

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

### **FRF? (Get Referencing Result)**

Description: Gets 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. 166), FPL (p. 167) or FRF (p. 168) or when the position was set directly with POS (p. 194) (depending on the referencing method selected with RON (p. 196)).

**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. 140), STP (p. 206), and HLT (p. 172).

Format: GOH [{<AxisID>}]

Arguments: <AxisID>: Is one axis of the controller; if not specified, all axes are affected.

Response: None

Troubleshooting: Illegal axis identifier

**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

Example:

For the C-663, the response to HDR? reads as follows:

```

#RecordOptions
0=Nothing is recorded
1=Commanded Position of Axis
2=Actual Position of 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
81=Analog input (Channel = 1 - 9)
#TriggerOptions
0=default setting
1=any command changing position (e.g., MOV)
2=next command
6=any command changing position (e.g., MOV), reset trigger after execution
#Additional information
4 record tables
1024 datapoints per table
end of help

```

## Notes:

TriggerOptions = 0 (default) means that recording is triggered by the STE command (p. 206).

Option 73 (Motor Output of Axis) is not relevant for the C-663 and only available for compatibility reasons.

**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. 140) and STP (p. 206) in contrast abort current motion as fast as possible for the controller without taking care of maximum velocity and acceleration.

Format: HLT [{<AxisID>}]

Arguments: <AxisID>: is one axis of the controller, if left out, all axes are stopped

Response: none

Troubleshooting: Illegal axis identifier

Notes: HLT stops motion with specified system deceleration regarding system inertia.

HLT stops all motion caused by motion commands (e.g., MOV (p. 191), MVR (p. 192), GOH (p. 170), STE (p. 206)), commands for referencing (FNL (p. 166), FPL (p. 167), FRF (p. 168)), and macros (MAC (p. 184)).

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

**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. 254) for further information.

Format: HPA?

Arguments: None

Response {<ParamID>}"=<string> LF}

where

<ParamID> is the ID of one parameter, hexadecimal format

<string> is a string which describes the corresponding parameter.

The string has the following format:

<CmdLevel>TAB<MaxItem>TAB<DataType>TAB<FunctionGroupDescription>TAB<ParameterDescription>[{TAB<PossibleValue>"=<ValueDescription>}]

where

<CmdLevel> is the command level that allows write access to the parameter value.

<MaxItem> is the maximum number of items of the same type which are affected by the parameter. With the C-663, 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 which the parameter belongs to.

<ParameterDescription> is the parameter name.

<PossibleValue> is one value from the permissible 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. 202) influences the parameter settings in volatile memory (RAM).

WPA (p. 219) copies parameter settings from volatile to nonvolatile memory.

SEP (p. 200) writes parameter settings directly into nonvolatile memory (without changing settings in volatile memory).

RPA (p. 197) resets volatile memory to the values from nonvolatile memory.

**HPV? (Get Parameter Value Description)**

Description: Responds with a help string that contains possible parameter values. Use HPA? instead to get a help string which contains all available parameters with short descriptions.

Format: HPV?

Arguments: None

Response: <string> has the following format:

```
"#Possible parameter values are:  
{<PamID> <ItemID> "=" <ListType>  
[ {TAB <PossibleValue> "=" <ValueDescription>} ] }  
#CCL levels are:  
{<PamID> <ItemID> "=" <CmdLevel>}  
#HPA_Category enabled  
end of help"
```

where

<PamID> is the ID of one parameter, hexadecimal format

<ItemID> is an element (axis, channel, whole system) of the controller, if item=0 the description applies to all elements

<ListType> determines how the possible parameter values listed in the string have to be interpreted:

- 0 = parameter not applicable for this element
- 1 = enumeration
- 2 = min/max

<PossibleValue> is a value from the permissible data range

<ValueDescription> is the meaning of the corresponding value

Some parameters are write protected (by a command level > 1) for certain elements. These parameters are listed below the "#CCL levels are" line.

<CmdLevel> is the command level that allows write access to the parameter value.

The "#HPA\_Category enabled" line is evaluated by the PC software for display purposes.

Notes: For C-663, the specifications

#Possible parameter values are:

and

#CCL levels are:

are left out of the response to HPV?, because all required information is already in the response to HPA?

### JAS? (Query Joystick Axis Status)

Description: Queries the current status of the specified axis of the specified joystick connected to the controller.

Format: JAS? [{<JoystickID> <JoystickAxis>}]

Arguments: <JoystickID> is one joystick connected to the controller; see below for details.

<JoystickAxis> is one of the axes of the joystick; see below for details.

Response: {<JoystickID> <JoystickAxis>}"=<Amplitude>}

where

<Amplitude> is the factor which is currently applied to the current valid velocity setting of the controlled motion axis, corresponds to the current displacement of the joystick axis. See below for details.

Notes: A joystick can be connected to the **Joystick** socket of the C-663, the identifier is 1. The C-663 supports one axis of the joystick, the identifier of the joystick axis is 1. Refer also to "Commandable Elements" (p. 18) for more information.

The <Amplitude> factor is applied to the velocity set with VEL (p. 217), the range is from -1.0 to 1.0. Examples: With a factor of 0, the joystick axis is at the center position; with a factor of -0.7, the displacement of the joystick axis is about 2/3 in negative direction, provided that a linear lookup table is currently valid (see JLT (p. 178) for an example).

**JAX (Set Axis Controlled By Joystick)**

Description: Sets axis controlled by a joystick which is connected to the controller.

Each axis of the controller can only be controlled by one joystick axis.

Format: JAX <JoystickID> <JoystickAxis> <AxisID>

Arguments: <JoystickID> is one joystick connected to the controller; see below for details.

<JoystickAxis> is one of the axes of the joystick; see below for details.

<AxisID> is one axis of the controller.

Response: none

Notes: A joystick can be connected to the **Joystick** socket of the C-663, the identifier is 1. The C-663 supports one axis of the joystick, the identifier of the joystick axis is 1. Refer also to "Commandable Elements" (p. 18) for more information.

**JAX? (Get Axis Controlled By Joystick)**

Description: Queries axis controlled by a joystick which is connected to the controller.

Format: JAX? [{<JoystickID>} <JoystickAxis>]

Arguments: <JoystickID> is one joystick connected to the controller; see below for details.

<JoystickAxis> is one of the axes of the joystick; see below for details.

Response: {<JoystickID>} <JoystickAxis> = "{<AxisID>}  
where  
<AxisID> is one axis of the controller.

Notes: A joystick can be connected to the **Joystick** socket of the C-663, the identifier is 1. The C-663 supports one axis of the joystick, the identifier of the joystick axis is 1. Refer also to "Commandable Elements" (p. 18) for more information.

**JBS? (Query Joystick Button Status)**

Description: Queries the current status of the specified button of the specified joystick connected to the controller.

Format: JBS? [{<JoystickID>} <JoystickButton>]}

Arguments: <JoystickID> is one joystick connected to the controller; see below for details.

<JoystickButton> is one of the buttons of the joystick; see below for details.

Response: {<JoystickID>} <JoystickButton> "="<State>}

where

<State> indicates if the joystick button is pressed; 0 = not pressed, 1 = pressed.

Notes: A joystick can be connected to the **Joystick** socket of the C-663, the identifier is 1. The C-663 supports one button of the joystick, the identifier of the joystick button is 1. Refer also to "Commandable Elements" (p. 18).

**JDT (Set Joystick Default Lookup Table)**

Description: Sets lookup table type for the specified axis of the specified joystick connected to the controller.

The current valid lookup-table content for the specified joystick axis is overwritten by the selection made with JDT.

Format: JDT {<JoystickID>} <JoystickAxis> <uint>}

Arguments: <JoystickID> is one joystick connected to the controller; see below for details.

<JoystickAxis> is one of the axes of the joystick; see below for details.

<uint> defines the type of lookup-table profile to use; see below for details.

Response: none

Notes: A joystick can be connected to the **Joystick** socket of the C-663, the identifier is 1. The C-663 supports one axis of the joystick, the identifier of the joystick axis is 1. Refer also to "Commandable Elements" (p. 18) for more information.

**Note: The number of write cycles in the nonvolatile memory is limited. Change the lookup table type only if necessary.**

Available lookup tables: The C-663 provides the following types for the lookup-table profile:

- 1 = linear (default)
- 2 = parabolic

### JLT (Fill Joystick Lookup Table)

Description: Fills the lookup table for the specified axis of the joystick specified that is connected to the controller.

The amplitudes of the joystick axes (i.e., their displacements) are mapped to the current valid velocity settings of the controller axes. For each joystick axis there is a lookup table that defines this mapping. With JLT this table can be written, or a default table profile provided by the controller can be loaded with the JDT command (p. 177).

Each lookup table consists of 256 points. By default, the first point corresponds to the maximum joystick axis displacement in negative direction, the 256th point to the maximum displacement in positive direction.

Format: `JLT <JoystickID> <JoystickAxis> <Addr> <floatn>`

Arguments: `<JoystickID>` is one joystick connected to the controller; see below for details.

`<JoystickAxis>` is one of the axes of the joystick; see below for details.

`<Addr>` is the index of a point in the lookup table, starts with 1.

`<floatn>` is the value of point n.

Response:

none

Notes:

A joystick can be connected to the **Joystick** socket of the C-663, the identifier is 1. The C-663 supports one axis of the joystick, the identifier of the joystick axis is 1. Refer also to "Commandable Elements" (p. 18) for more information.

The values <floatn> are factors applied during joystick control to the velocity set with VEL (p. 217), the range is from -1.0000 to 1.0000.

The values <floatn> are automatically stored in the non-volatile memory of the C-663.

Example: Point 1 has the value -1 in the current lookup table and therefore the controlled axis will move with full velocity in a negative direction at the maximum negative displacement of the joystick. Points 124 to 133 have a value of 0, i.e., at the center position of the joystick and in a small area around the center, the velocity is 0 and the controlled axis will not move. Point 236 has the value 0.8369, i.e., when the displacement of the joystick axis is about 2/3 in positive direction, the controlled axis will move in a positive direction with about 4/5 of the full velocity. Point 256 has the value 1, i.e., the controlled axis will move with full velocity in positive direction at the maximum positive displacement of the joystick.

**Note: The number of write cycles in the nonvolatile memory is limited. Write values to the lookup table only if necessary.**

### JLT? (Get Joystick Lookup Table Values)

Description:

Queries the current valid lookup table values.

Format:

JLT? [<StartPoint> <NumberOfPoints> [{<JoystickID> <JoystickAxis>}]]

Arguments:

<StartPoint>: is the start point in the lookup table, starts with 1

<NumberOfPoints>: is the number of points to be read per joystick axis; maximum number is 256.

<JoystickID> is one joystick connected to the controller; see below for details.

<JoystickAxis> is one of the axes of the joystick; see below for details.

Response: The lookup table content in GCS array format, see the separate manual for GCS array, SM 146E, and the example below.

Notes: With the C-663, <JoystickID> and <JoystickAxis> must be left out in the JLT? command, but <StartPoint> and <NumberOfPoints> are always required.

The values <floatn> in the lookup table are factors that are applied to the velocity set with VEL (p. 217), the range is -1.0000 to 1.0000.

Example:

```
jlt? 1 20
# TYPE = 1
#
# SEPARATOR = 32
# DIM = 1
# NDATA = 20
# NAME0 = Joysticktable 1
# END HEADER
-1.0000
-0.9922
-0.9834
-0.9756
-0.9678
-0.9590
-0.9512
-0.9434
-0.9346
-0.9268
-0.9189
-0.9102
-0.9023
-0.8945
-0.8857
-0.8779
-0.8701
-0.8613
-0.8535
-0.8457
```

**JON (Set Joystick Activation Status)**

- Description: Activates or deactivates a joystick connected to the controller.
- Format: JON {<JoystickID> <uint>}
- Arguments: <JoystickID> is one joystick connected to the controller; see below for details.  
<uint> 1 activates the joystick, 0 deactivates the joystick.
- Response: none
- Notes: A joystick can be connected to the **Joystick** socket of the C-663, the identifier is 1. For more information, see "Connecting an Analog Joystick" (p. 54).  
  
Before a joystick can be activated with JON, its axes must have been assigned to the controller axes using JAX (p. 176).  
  
When a joystick connected to the C-663 is activated with the JON command, this joystick controls the axis velocity ("commanded velocity" output by the profile generator). During joystick control, the soft limit specified by the parameter 0x15 or 0x30 is set as the target position. Details on the parameters can be found in "Travel Range and Soft Limits" (p. 34). When disabling a joystick, the target position is set to the current position of the joystick-controlled axis.  
  
Motion commands such as MOV (p. 191) are not allowed when a joystick is active for the axis. For further information, see "Joystick Control" (p. 102).

**JON? (Get Joystick Activation Status)**

- Description: Queries the activation state of the specified joystick connected to the controller.
- Format: JON? [{<JoystickID>}]
- Arguments: <JoystickID> is one joystick connected to the controller; see below for details.

Response: {<JoystickID>}"=<uint>}

where

<uint> is the joystick activation state: 1 = joystick activated, 0 = joystick deactivated.

Notes: A joystick can be connected to the **Joystick** socket of the C-663, the identifier is 1. For more information, see "Connecting an Analog Joystick" (p. 54).

### JRC (Jump Relatively Depending On Condition)

Description: Jumps relatively depending on a specified condition of the following type: one specified value is compared with a queried value according to a specified rule.

Can only be used in macros.

Format: JRC <Jump> <CMD?> <OP> <Value>

Arguments: <Jump> is the size of the relative jump. -1 means the macro execution pointer jumps back to the previous line, 0 means the command is executed again, which is the same behavior as with WAC (p. 218). 1 jumps to the next line, making the command unnecessary, and 2 jumps over the next command. Only jumps within the current macro are allowed.

<CMD?> is one query command in its usual notation. The response has to be a single value and not more. For an example see below.

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

= <= <> >= !=

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

<Value> is the value to be compared with the response to <CMD?>.

Response: None

Troubleshooting: Check proper jump target

Example: Using the following macro, you can stop motion of axis "1" using a stop button connected to a digital input. The stop button is checked until the axis has reached the target position (query ONT?). When the stop button is pressed before the target position has been reached: The response

to the POS? query is copied into the TARGET variable. This variable is then used as second argument for the MOV command. Therefore, the positioner just stays where it was. To clean up, TARGET is defined as empty with the VAR command which deletes the variable.

Write the "stop" macro:

```
MAC BEG stop
MOV 1 20
JRC 2 DIO? 1 = 1
JRC -1 ONT? 1 = 0
CPY TARGET POS? 1
MOV 1 ${TARGET}
VAR TARGET
MAC END
```

### 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-663 firmware detects the presence or absence of limit switches using a parameter (ID 0x32). The C-663 activates or deactivates the stop motion function at the limit switches according to the value of this parameter as well as reference moves using FNL (p. 166) or FPL (p. 167).

Adapt the parameter value to your hardware using SPA (p. 202) or SEP (p. 200). For further information, see "Limit Switch Detection" (p. 33).

You can use the digital input lines instead of the limit switches as source of the negative or positive limit switch signal. For further information, see "Digital Input Signals" (p. 96).

**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 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 <macro name>

Starts recording a macro to be named *macro name* 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?

Reports the last error that occurred while the macro was running.

Response: <macro name> <uint1>"=<uint2>  
<"<"CMD">">

where <macro name> 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 <macro name>

Sets specified macro as startup macro. This macro will be run automatically after the next switch-on or reboot of the controller. If <macro name> is not specified, the current startup macro selection is canceled.

MAC DEF?

Asks for the startup macro

Response: <macro name>

If a startup macro is not defined, the response is an empty string with the terminating character.

**MAC DEL <macro name>**

Deletes specified macro.

**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. 132) 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. For further information, see "Working with Macros" (p. 112) and "Target and Sender Address" (p. 132).

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. For further information, see "Variables" (p. 132).

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-663 can only be used in macros:

DEL (p. 154), JRC (p. 182), MEX (p. 189) and WAC (p. 218).

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

Any 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. 140) and STP (p. 206).

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. 140) 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? [<macro name>]

Arguments <macro name>: 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 <macro name> was specified, <string> is the content of this macro;

If <macro name> was not specified, <string> is a list with the names of all stored macros

Troubleshooting: Macro <macro name> not found

### **MAN? (Get Help String For Command)**

Description: Shows a detailed help text for individual commands.

Format: MAN? <CMD>

Arguments: <CMD> is the command mnemonic of the command for which the help text is to be displayed (see below).

Response: A string that describes the command.

Notes: A detailed help text can be displayed for the following GCS commands:

CTO, CTO?, WPA

Example: Send: MAN? CTO

Receive:

```
CTO {<TrigOutID> <CTOPam> <Value>} Set
Configuration Of Trigger Output
#AvailableCTOparameters
<CTOPam> <Description>
<CTOPam> (configuration parameter) :
1 TriggerStep
2 Axis
3 TriggerMode
7 Polarity
8 StartThreshold
9 StopThreshold
10 TriggerPosition
#AvailableTriggerModes
<Value> <Description>
0 PositionDistance
2 OnTarget
5 MotionError
6 InMotion
7 Position+Offset
8 SinglePosition
#AvailablePolarities
```

```
<Value> <Description>
0 ActiveLow
1 ActiveHigh
end of help
```

### MAT (Calculate And Save To Variable)

Description: Carries out a mathematical operation or bit operation and saves the result as a variable (p. 132).

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. 218).

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

Example: Send: MAC START LOOP

Note:

LOOP macro contains the following:

```
MAC START KEY1
MAC START KEY2
MEX DIO? 4 = 1
MAC START LOOP
```

KEY1 macro contains the following:

```
MEX DIO? 4 = 1
MEX DIO? 1 = 0
MVR 1 1.0
DEL 100
```

KEY2 macro contains the following:

```
MEX DIO? 4 = 1
MEX DIO? 2 = 0
MVR 1 -1.0
DEL 100
```

LOOP macro forms an infinite loop by permanently calling KEY1, KEY2 and itself.

KEY1 checks the state of the digital input channel 1 (is on the I/O socket (p. 281)). If it is not set (0), the macro is aborted, otherwise the macro will move axis 1 by 1.0 in positive direction (relative move).

KEY2 checks the state of the digital input channel 2 and moves axis 1 in negative direction accordingly.

By connecting the digital input channels 1, 2 and 4 with pushbuttons, e.g., with the C-170.PB pushbutton box, it is possible to realize interactive control of an axis without any software assistance. The delay (DEL 100) is required to avoid generating multiple MVR commands while pressing the pushbutton for a short time.

Channel 4 is used as a global exit. Since MEX only stops execution of the current macro, it must also be included in the calling macro, which would otherwise continue.

**MOV (Set Target Position)**

Description: Sets an absolute target position for the specified axis.

Format: MOV {<AxisID> <Position>}

Arguments: <AxisID> is one axis of the controller.  
<Position> is the absolute target position in physical units.

Response: none

Notes: The target position must be within the soft limits. Use TMN? (p. 210) and TMX? (p. 211) to query the current valid travel range.  
The motion can be stopped by #24 (p. 140), STP (p. 206), and HLT (p. 172).

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.

Motion commands such as MOV are not allowed when a joystick is active for the axis. For further information, see "Joystick Control" (p. 102).

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. 164) 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

Notes: The target position can be changed by commands that cause motion (e.g., MOV (p. 191), MVR (p. 192), GOH (p. 170), STE (p. 206)) or by the joystick (when disabling a joystick, the target position is set to the current position for joystick-controlled axes).

### **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 target position must be within the soft limits. Use TMN? (p. 210) and TMX? (p. 211) to get the currently valid soft limits, and MOV? (p. 191) to get the current target.

The motion can be stopped by #24 (p. 140), STP (p. 206), and HLT (p. 172).

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.

Motion commands such as MVR are not allowed when a joystick is active for the axis. For further information, see "Joystick Control" (p. 102).

Example:

Send: <code>MOV 1 0.5</code>	Note: This is an absolute motion.
Send: <code>POS? 1</code>	Receive: <code>1=0.500000</code>
Send: <code>MOV? 1</code>	Receive: <code>1=0.500000</code>
Send: <code>MVR 1 2</code>	Note: This is a relative motion.
Send: <code>POS? 1</code>	Receive: <code>1=2.500000</code>
Send: <code>MVR 1 2000</code>	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: <code>MOV? 1</code>	Receive: <code>1=2.500000</code>
Send: <code>POS? 1</code>	Receive: <code>1=2.500000</code>

### ONT? (Get On-Target State)

Description: Gets the on-target state of the specified axis.

If all arguments are left out, gets 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 on-target state is influenced by the settings for the settling window (parameter 0x36) and the delay time (parameter 0x3F). For details, see "On-Target State" (p. 32).

**POS (Set Real Position)**

Description:	Sets the current position of the axis (does not cause motion).
Format:	POS {<AxisID> <Position>}
Arguments:	<AxisID> is one axis of the controller. <Position> is the new current position in physical units.
Response:	None
Troubleshooting:	Illegal axis identifier
Notes:	It is only possible to set the current position with POS when the referencing method "0" is selected, see RON (p. 196).  An axis is considered as "referenced" when the position has been set with POS (for more information, see "Referencing" (p. 38)).
	The minimum and maximum commandable positions (TMN? (p. 210), TMX? (p. 211)) are not adapted when a position is set with POS. This can result in target positions allowed by the C-663 but cannot be approached by the hardware. Target positions are also possible that can be approached by the hardware but are denied by the C-663. Furthermore, the zero position can be outside of the physical travel range after using POS.  If there is a sensor for the axis: In closed-loop operation, POS also modifies the current sensor position, which can be requested with TSP? (p. 214) and set with TSP (p. 213).

**POS? (Get Real Position)**

Description:	Queries the current axis position.
	If no arguments are specified, the current position of all axes is queried.
Format:	POS? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller.

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

where

<float> is the current axis position in physical units.

Troubleshooting: Illegal axis identifier

Notes: The content of the response to POS? depends on the current operating mode (p. 20) of the C-663:

- Closed-loop operation: The response to POS? is based on the position measured by the sensor.
- Open-loop operation: The answer to POS? is based on the commanded position (output of the profile generator). If a position sensor is present: The command TSP? (p. 214) queries the measured position. For the adjustment of POS? and TSP?, see "Open-Loop Operation with Sensor: Position Adjustment" (p. 83).

### 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: {<macro name> LF}

where

<macro name> 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. 194) or a reference move can be started with FRF (p. 168), FNL (p. 166) or FPL (p. 167). Relative motion with MVR (p. 192) 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. Axis motion is only possible after the reference move.

For further information, see "Referencing" (p. 38) and "Travel Range and Soft Limits" (p. 34).

### **RON? (Get Reference Mode)**

Description: Gets 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

Note: Further information can be found in the description of the RON command (p. 196).

### 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. 172) you can obtain a list of the available parameters. SPA (p. 202) influences the parameter settings in volatile memory, WPA (p. 219) writes parameter settings from volatile to nonvolatile memory, and SEP (p. 200) writes parameter settings directly into nonvolatile memory (without changing the settings in volatile memory).

See SPA for an example.

Format: RPA [{<ItemID> <PamID>}]

Arguments: <ItemID> is the element for resetting a parameter. See below for details.

<PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details.

Response: none

Troubleshooting: Illegal element identifier, wrong parameter ID

Notes: With the C-663, you can reset either all parameters or one single parameter with RPA.

**Available element IDs and parameter IDs:** An element is an axis (the identifier can be changed with SAI (p. 199)) or the entire system. Refer to "Commandable Elements" (p. 18) for further information.

Valid parameter IDs are specified in "Parameter Overview" (p. 254).

### **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:** The duration of the recording can be calculated as follows:

Rec. duration = cycle time of the servo loop \* RTR value \* number of points

where

the cycle time of the servo loop for the C-663 is 50 µs

the number of points for the C-663 is 1024 (length of data recorder table)

For further information, see "Data Recorder" (p. 87).

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

### **RTR? (Get Record Table Rate)**

**Description:** Gets 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 identifier can consist of up to 8 characters. Use TVI? (p. 215) to get valid characters.

The new axis identifier is only stored in the volatile memory of the C-663. A changed axis identifier can be stored permanently in the C-663 with the WPA (p. 219) command.

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

Description: Queries the axis identifiers.

Refer also to "Commandable Elements" (p. 18).

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.

**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. 197) to activate them (write them to volatile memory) without controller reboot.

**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. 172) returns a list of the available parameters.

SPA (p. 202) writes parameter settings into volatile memory (without changing the settings in nonvolatile memory).

WPA (p. 219) writes parameter settings from volatile to nonvolatile memory.

Format: SEP <Pswd> {<ItemID> <PamID> <PamValue>}

Arguments <Pswd> is the password for writing to the nonvolatile memory; the default value is "100".

<ItemID> is the element for changing a parameter in the nonvolatile memory. See below for details.

<PamID> is the parameter identifier, can be written in hexadecimal or decimal format. See below for details.

<PamValue> is the value for setting the specified parameter of the specified element.

Response: None

Troubleshooting: Illegal element identifier, wrong parameter ID, invalid password

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

With the C-663 you can only write one parameter per SEP

command.

- |                                       |  |
|---------------------------------------|--|
| Available item IDs and parameter IDs: | An element is an axis (the identifier can be changed with SAI (p. 199)) or the entire system. Refer to "Commandable Elements" (p. 18) for further information. |
|                                       | Valid parameter IDs are specified in "Parameter Overview" (p. 254).  |

### **SEP? (Get Nonvolatile Memory Parameters)**

- |  |   |
|--|---|
| Description:                             | Gets the value of a parameter of a specified element from nonvolatile memory.<br><br>With HPA? (p. 172) you can obtain a list of the available parameters and their IDs.  |
| Format:                                  | SEP? [{<ItemID> <PamID>}]   |
| Arguments:                               | <ItemID> is the element for querying a parameter value from nonvolatile memory. See below for details.<br><br><PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details.                        |
| Response:                                | {<ItemID> <PamID>}=<PamValue> LF<br><br>where<br><br><PamValue> is the value of the specified parameter for the specified element   |
| Troubleshooting:                         | Illegal element identifier, wrong parameter ID  |
| Notes:                                   | With the C-663, you can query either all parameters or one single parameter per SEP? command.   |
| Available element IDs and parameter IDs: | An element is an axis (the identifier can be changed with SAI (p. 199)) or the entire system. Refer to "Commandable Elements" (p. 18) for further information.<br><br>Valid parameter IDs are specified in "Parameter Overview" (p. 254). |

### **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. 197).

**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. 172) returns a list of the available parameters.

SEP (p. 200) writes parameter settings directly into nonvolatile memory (without changing the settings in volatile memory).

WPA (p. 219) writes parameter settings from volatile to nonvolatile memory.

RPA resets volatile memory to the value in nonvolatile memory.

Troubleshooting: Illegal element identifier, wrong parameter ID, value out of range

Notes: With the C-663, you can write only one parameter per SPA command.

Available element IDs and parameter IDs: An element is an axis (the identifier can be changed with SAI (p. 199)) or the entire system. Refer to "Commandable Elements" (p. 18) for further information.

Valid parameter IDs are specified in the parameter overview (p. 254).

Example 1: Send: SPA 1 0x3111 100

Note: Sets the P term 1 of the servo algorithm for axis 1 to 100, the parameter ID is written in hexadecimal format

Send: SPA 1 12561 150

Note: Sets the P term 1 of the servo algorithm for axis 1 to 150, the parameter ID is written in decimal format

Example 2:

The P, I and D parameters of the servo algorithm must be adapted to a new load applied to the mechanics connected.

Send: SPA 1 0x3111 150

Note: The P term 1 is set to 150 for axis 1. The setting is made in volatile memory only.

Use SPA to set the I and D terms in volatile memory and then test the function 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.

Send: WPA 100

Note: Refer to the command description for WPA (p. 219) for details on the extent of the saved settings.

Example 3:

Send: SEP 100 LEFT 0xA 20

Note: The maximum velocity must be set to 20 mm/s for 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. 172).

**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-663, 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. 199)) or the entire system. For further information, see "Commandable Elements" (p. 18).

Valid parameter IDs can be found in the parameter overview (p. 254).

### **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. 138) which should be preferred when the controller is doing 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	Is referencing	In motion	Motor on	-	-	-	Error flag

Bit	7	6	5	4	3	2	1	0
Description	Digital input line 4	Digital input line 3	Digital input line 2	Digital input line 1	-	Positive limit switch	Reference switch	Negative limit switch

Example: Send: SRG? 1 1

Receive: 1 1=0x9002

Note: The response is in hexadecimal format. It means that axis 1 is on target (on-target state = true), the motor is activated for that axis, no error has occurred, the states of digital input lines 1 to 4 are low, and axis 1 is on the positive side of the reference switch.

**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. 158).
- The recorded data can be read with the DRR? command (p. 161).
- Format: STE <AxisID> <Amplitude>
- Arguments: <AxisID> is one axis of the controller  
<Amplitude> is the size of the step. See below for details.
- Response: None
- Troubleshooting: The target position must be within the soft limits. Use TMN? (p. 210) and TMX? (p. 211) to get the currently valid soft limits, and MOV? (p. 191) to get the current target.  
Motion commands such as STE are not allowed when the joystick is active for the axis. For further information, see "Joystick Control" (p. 102).
- Notes: A "step" consists of a relative move of the specified amplitude which is done 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. 140).
- Format: STP
- Arguments: None
- Response: None
- Troubleshooting: Communication breakdown

**Notes:**

STP stops all motion caused by motion commands (e.g., MOV (p. 191), MVR (p. 192), GOH (p. 170), STE (p. 206)), commands for referencing (FNL (p. 166), FPL (p. 167), FRF (p. 168)), and macros (MAC (p. 184)). Also stops macro running.

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

HLT (p. 172) in contrast to STP stops motion with specified system deceleration regarding system inertia.

**SVO (Set Motor State)**

**Description:** Activates / deactivates the motor for the specified axes.

**Format:** SVO {<AxisID> <MotorState>}

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

<MotorState> can have the following values:

0 = Motor activated

1 = Motor deactivated

**Response:** None

**Troubleshooting:** Illegal axis identifier

**Notes:** The motor must be activated so that motion can be commanded.

When the motor is activated, the target position is set to the current position in order to avoid jumps in the mechanics.

Before changing the operating mode (closed-loop / open-loop operation, see "Setting the Operating Mode" (p. 82)), the motor must be deactivated.

A startup macro can be used to configure the controller so that the motor is automatically activated after it is switched on or rebooted. For more information, see "Setting up a Startup Macro" (p. 119).

If the axis has a brake, activating the motor with SVO influences the brake's activation state:

- Activating the motor deactivates the brake.
- Deactivating the motor activates the brake. If the motor is deactivated, the brake can be activated or deactivated with BRA (p. 145). Secure the positioner against moving unintentionally before you deactivate the brake before you deactivate the brake with BRA!

If motion error occurs the motor is deactivated and the brake is activated. For more information see "Behavior with Motion Errors" (p. 85).

### **SVO? (Get Motor State)**

Description: Queries the activation state of the motor for the specified axes.

If no arguments are specified, the activation state of the motor is queried for all axes.

Format: SVO? [{<AxisID>}]

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

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

where

<MotorState> is the current activation state of the motor:  
 0 = Motor deactivated (axis cannot be moved)  
 1 = Motor activated (axis can be moved)

Troubleshooting: Illegal axis identifier

### **TAC? (Tell Analog Channels)**

Description: Gets the number of installed analog lines.

Format: TAC?

Arguments: None

Response: <uint> indicates the total number of analog lines (inputs and outputs).

Notes: Gets the number of analog input lines on the I/O socket (p. 281) of the C-663 (Input 1 to Input 4). Note that these lines can also be used for digital input. For further information, refer to "Commandable Elements" (p. 18).

**TAV? (Get Analog Input Voltage)**

Description: Get voltage at analog input.

Format: TAV? [{<AnalogInputID>}]

Arguments: <AnalogInputID> is the identifier of the analog input channel; see below for details.

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

where

<float> is the current voltage at the analog input in volts

Notes: Using the TAV? command, you can directly read the Input 1 to Input 4 lines on the I/O socket (p. 281) of the C-663. The identifiers of the lines are 1 to 4. Refer to "Commandable Elements" (p. 18) for further information.

You can record the values of the analog input lines using the DRC record option 81 (p. 158).

**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.

**TIO? (Tell Digital I/O Lines)**

Description: Tells number of installed digital I/O lines

Format: TIO?

Arguments: none

Response: I=<uint1>  
O=<uint2>

where

<uint1> is the number of digital input lines.  
<uint2> is the number of digital output lines.

Notes: The digital output lines reported by TIO? are Output 1 to Output 4. The states of the Output 1 to Output 4 lines can be set using the DIO command (p. 157). Furthermore, you can program the Output 1 to Output 4 lines using the CTO command (p. 149) (trigger configuration) and the TRO command (p. 211) (trigger activation/deactivation).

The digital input lines reported by TIO? are Input 1 to Input 4. They can be read with DIO? (p. 158), #4 (p. 138) and SRG? (p. 204).

All the lines are located on the **I/O** socket (p. 281) of the C-663.

### **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. 155) 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. 155) command, the maximum commandable position is automatically adapted to the new zero position.

**TNR? (Get Number of Record Tables)**

Description: Gets 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-663 has four data recorder tables with 1024 data points per table.

For further information, see "Data Recorder" (p. 87).

**TRO (Set Trigger Output State)**

Description: Activates or deactivates the trigger output conditions set with CTO (p. 149) for the specified digital output line.

Format: TRO {<TrigOutID> <TrigMode>}

Arguments: <TrigOutID> is a digital output line of the controller; see below for further details.

<TrigMode> can have the following values:  
 0 = Trigger output deactivated  
 1 = Trigger output activated

Response: None

Troubleshooting: Illegal identifier of the digital output line

Notes: <TrigOutID> corresponds to the digital output lines Output 1 to Output 4, IDs = 1 to 4; for further information, see "I/O" (p. 281).

Do not use DIO (p. 157) on digital output lines where the trigger output is activated by TRO.

### **TRO? (Get Trigger Output State)**

Description: Queries the activation status of the trigger output configuration made with CTO (p. 149) for the specified digital output line.

If no arguments are specified, queries state of all digital output lines.

Format: TRO? [{<TrigOutID>}]

Arguments: <TrigOutID> is one digital output line of the controller, see TRO (p. 211) for more details.

Response: {<TrigOutID>}"=<TrigMode> LF}

where

<TrigMode> is the current state of the digital output line:  
 0 = Trigger output deactivated  
 1 = Trigger output activated

Troubleshooting: Illegal identifier of the digital output line

**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: The C-663 firmware detects the presence or absence of a reference switch via a parameter (ID 0x14). The C-663 activates or deactivates reference moves to the reference switch (FRF command (p. 168)) according to the value of this parameter. Adapt the parameter value to your hardware using SPA (p. 202) or SEP (p. 200). For further information, see "Reference Switch Detection" (p. 33).

You can use a digital input line instead of the reference switch as source of the reference point signal for the FRF command. For further information, see "Digital Input Signals" (p. 96).

**TSP (Set Input Signal Position Value)**

Description: Sets the current position of the specified input signal channel (does not cause motion).

Format: TSP {<InputSignalID>} <Position>

Arguments: <InputSignalID> is one input signal channel of the controller

<Position> is the new current position in physical units

Response: None

Notes: The input signal channel of the C-663 represents the position sensor of the axis; for details, refer to

"Commandable Elements" (p. 18).

Setting the current sensor position with TSP is only possible if the motor is deactivated with SVO (p. 207).

Setting the current sensor position with TSP does **not** change the current axis position, which can be queried with POS? (p. 194)

When referencing is done for an axis, the current sensor position is set to a defined value **only in closed-loop** operation. After referencing in **open-loop** operation, the sensor position should therefore be matched with TSP to the current axis position, see "Open-Loop Operation with Sensor: Position Adjustment" (p. 83).

### TSP? (Get Input Signal Position Value)

Description: Queries the current position of the specified input signal channel.

If no arguments are specified, the current position of all input signal channels is queried.

Format: `TSP? [{<InputSignalID>}]`

Arguments: `<InputSignalID>` is one input signal channel of the controller

Response: `{<InputSignalID>}=<float> LF`

where

`<float>` is the current position of the input signal channel, in physical units

Notes: The input signal channel of the C-663 represents the position sensor of the axis; Refer to "Commandable Elements" (p. 18) for details.

See TSP (p. 213) for more information.

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

**Description:** Returns a string with characters which can be used for axis identifiers.

Use SAI (p. 199) to change the axis identifiers and SAI? (p. 199) to ask for the current valid axis identifiers.

**Format:** TVI?

**Arguments:** None

**Response:** <string> is a list of characters

**Notes:** With the C-663, the string consists of  
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ-\_

**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. 132) 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. 132) 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. 142) for another example.

### **VAR? (Get Variable Values)**

Description: Gets values of variables.

If VAR? is combined with CPY (p. 147), JRC (p. 182), MEX (p. 189) or WAC (p. 218), the response to VAR? has to be a single value and not more.

Refer to "Variables" (p. 132) 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. 132) 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. 132) for details regarding local and global variables.

Example: See ADD (p. 142) 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 identifier

Notes: 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 parameter ***Closed-Loop Velocity (Phys. Unit/s)*** (0x49) in the volatile memory of C-663. The parameter value can be stored as default with WPA (p. 219), see "Adapting Settings" (p. 245) for details.

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

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

Description: Gets the versions of the firmware of the C-663 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. 189).

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. 197) is used to restore the parameters.

You can obtain a list of all available parameters with HPA? (p. 172).

Use SPA? (p. 202) to check the current parameter settings in volatile memory.

See SPA (p. 202) 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 where a parameter is to be saved from the volatile to the nonvolatile memory. See below for details.

<PamID> is the parameter identifier, can be written in hexadecimal or decimal format. See below for details.

**Response:** None

Troubleshooting:	Illegal element identifier, wrong parameter ID, invalid password
Notes:	Parameters can be changed in volatile memory with SPA (p. 202), ACC (p. 141), DEC (p. 153), and VEL (p. 217).
<p>When WPA is used without specifying any arguments except the password, the currently valid values of all parameters affected by the specified password are saved. Otherwise only one single parameter can be saved per WPA command.</p> <p><b>Note that the number of write cycles in the nonvolatile memory is limited. Write default settings only if necessary.</b></p>	
Valid passwords:	The password for writing to the nonvolatile memory is "100".
Available element IDs and parameter IDs:	An element is an axis (the identifier can be changed with SAI (p. 199)) or the entire system. Refer to "Commandable Elements" (p. 18) for further information.
	Valid parameter IDs are specified in "Parameter Overview" (p. 254).

## 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_	Open-loop commands (SVA,

	SERVO_ON	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_ENABLE	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_CHANNEL	VEL command not allowed in DTR Command Mode
214	PI_CNTR_POSITION_UNKNOWN	Position calculations failed
215	PI_CNTR_CONN_POSSIBLY_BROKEN	The connection between controller and stage may be broken
216	PI_CNTR_ON_LIMIT_SWITCH	The connected stage has driven into a limit switch, some controllers need CLR to resume operation
217	PI_CNTR_UNEXPECTED_STRUT_STOP	Strut test command failed because of an unexpected strut stop
218	PI_CNTR_POSITION_BASED_ON_ESTIMATION	While MOV! is running position can only be estimated!
219	PI_CNTR_POSITION_BASED_ON_INTERPOLATION	Position was calculated during MOV motion
220	PI_CNTR_INTERPOLATION_FIFO_UNDERRUN	FIFO buffer underrun during interpolation
221	PI_CNTR_INTERPOLATION_FIFO_OVERFLOW	FIFO buffer overflow during interpolation
230	PI_CNTR_INVALID_HANDLE	Invalid handle
231	PI_CNTR_NO BIOS FOUND	No bios found
232	PI_CNTR_SAVE_SYS_CFG_FAILED	Save system configuration

		failed
233	PI_CNTR_LOAD_SYS_CFG_FAILED	Load system configuration failed
301	PI_CNTR_SEND_BUFFER_OVERFLOW	Send buffer overflow
302	PI_CNTR_VOLTAGE_OUT_OF_LIMITS	Voltage out of limits
303	PI_CNTR_OPEN_LOOP_MOTION_SET_WHEN_SERVO_ON	Open-loop motion attempted when servo ON
304	PI_CNTR RECEIVING_BUFFER_OVERFLOW	Received command is too long
305	PI_CNTR_EEPROM_ERROR	Error while reading/writing EEPROM
306	PI_CNTR_I2C_ERROR	Error on I2C bus
307	PI_CNTR RECEIVING_TIMEOUT	Timeout while receiving command
308	PI_CNTR_TIMEOUT	A lengthy operation has not finished in the expected time
309	PI_CNTR_MACRO_OUT_OF_SPACE	Insufficient space to store macro
310	PI_CNTR_EUI_OLDVERSION_CFGDATA	Configuration data has old version number
311	PI_CNTR_EUI_INVALID_CFGDATA	Invalid configuration data
333	PI_CNTR_HARDWARE_ERROR	Internal hardware error
400	PI_CNTR_WAV_INDEX_ERROR	Wave generator index error
401	PI_CNTR_WAV_NOT_DEFINED	Wave table not defined
402	PI_CNTR_WAV_TYPE_NOT_SUPPORTED	Wave type not supported
403	PI_CNTR_WAV_LENGTH_EXCEEDS_LIMIT	Wave length exceeds limit
404	PI_CNTR_WAV_PARAMETER_NR	Wave parameter number error
405	PI_CNTR_WAV_PARAMETER_OUT_OF_LIMIT	Wave parameter out of range
406	PI_CNTR_WGO_BIT_NOT_SUPPORTED	WGO command bit not supported
500	PI_CNTR_EMERGENCY_STOP_BUTTON_ACTIVATED	The \"red knob\" is still set and disables system
501	PI_CNTR_EMERGENCY_STOP_BUTTON_WAS_ACTIVATED	The \"red knob\" was activated and still disables system - reanimation required
502	PI_CNTR_REDUNDANCY_LIMIT_EXCEEDED	Position consistency check failed
503	PI_CNTR_COLLISION_SWITCH_ACTIVATED	Hardware collision sensor(s) are activated

504	PI_CNTR_FOLLOWING_ERROR	Strut following error occurred, e.g. caused by overload or encoder failure
505	PI_CNTR_SENSOR_SIGNAL_INVALID	One sensor signal is not valid
506	PI_CNTR_SERVO_LOOP_UNSTABLE	Servo loop was unstable due to wrong parameter setting and switched off to avoid damage.
507	PI_CNTR_LOST_SPI_SLAVE_CONNECTION	Digital connection to external SPI slave device is lost
508	PI_CNTR_MOVE_ATTEMPT_NOT_PERMITTED	Move attempt not permitted due to customer or limit settings
509	PI_CNTR_TRIGGER_EMERGENCY_STOP	Emergency stop caused by trigger input
530	PI_CNTR_NODE_DOES_NOT_EXIST	A command refers to a node that does not exist
531	PI_CNTR_PARENT_NODE_DOES_NOT_EXIST	A command refers to a node that has no parent node
532	PI_CNTR_NODE_IN_USE	Attempt to delete a node that is in use
533	PI_CNTR_NODE_DEFINITION_IS_CYCLIC	Definition of a node is cyclic
536	PI_CNTR_HEXAPOD_IN_MOTION	Transformation cannot be defined as long as Hexapod is in motion
537	PI_CNTR_TRANSFORMATION_TYPE_NOT_SUPPORTED	Transformation node cannot be activated
539	PI_CNTR_NODE_PARENT_IDENTICAL_TO_CHILD	A node cannot be linked to itself
540	PI_CNTR_NODE_DEFINITION_INCONSISTENT	Node definition is erroneous or not complete (replace or delete it)
542	PI_CNTR_NODES_NOT_IN_SAME_CHAIN	The nodes are not part of the same chain
543	PI_CNTR_NODE_MEMORY_FULL	Unused nodes must be deleted before new nodes can be stored
544	PI_CNTR_PIVOT_POINT_FEATURE_NOT_SUPPORTED	With some transformations pivot point usage is not supported
545	PI_CNTR_SOFTLIMITS_INVALID	Soft limits invalid due to changes in coordinate system
546	PI_CNTR_CS_WRITE_PROTECTED	Coordinate system is write protected

547	PI_CNTR_CS_CONTENT_FROM_CONFIGFILE	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_OR	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
1024	PI_CNTR_MOTION_ERROR	Motion error: position error too large, servo is switched off automatically
1025	PI_CNTR_MAX_MOTOR_OUTPUT_REACHED	Maximum motor output reached

1063	PI_CNTR_EXT_PROFILE_UNALLOWED_CMD	User Profile Mode: Command is not allowed, check for required preparatory commands
1064	PI_CNTR_EXT_PROFILE_EXPECTING_MOTION_ERROR	User Profile Mode: First target position in User Profile is too far from current position
1065	PI_CNTR_PROFILE_ACTIVE	Controller is (already) in User Profile Mode
1066	PI_CNTR_PROFILE_INDEX_OUT_OF_RANGE	User Profile Mode: Block or Data Set index out of allowed range
1071	PI_CNTR_PROFILE_OUT_OF_MEMORY	User Profile Mode: Out of memory
1072	PI_CNTR_PROFILE_WRONG_CLUSTER	User Profile Mode: Cluster is not assigned to this axis
1073	PI_CNTR_PROFILE_UNKNOWN_CLUSTER_IDENTIFIER	Unknown cluster identifier
1090	PI_CNTR_TOO_MANY_TCP_CONNECTIONS_OPEN	There are too many open tcpip connections
2000	PI_CNTR_ALREADY_HAS_SERIAL_NUMBER	Controller already has a serial number
4000	PI_CNTR_SECTOR_ERASE_FAILED	Sector erase failed
4001	PI_CNTR_FLASH_PROGRAM_FAILED	Flash program failed
4002	PI_CNTR_FLASH_READ_FAILED	Flash read failed
4003	PI_CNTR_HW_MATCHCODE_ERROR	HW match code missing/invalid
4004	PI_CNTR_FW_MATCHCODE_ERROR	FW match code missing/invalid
4005	PI_CNTR_HW_VERSION_ERROR	HW version missing/invalid
4006	PI_CNTR_FW_VERSION_ERROR	FW version missing/invalid
4007	PI_CNTR_FW_UPDATE_ERROR	FW update failed
4008	PI_CNTR_FW_CRC_PAR_ERROR	FW Parameter CRC wrong
4009	PI_CNTR_FW_CRC_FW_ERROR	FW CRC wrong
5000	PI_CNTR_INVALID_PCC_SCAN_DATA	PicoCompensation scan data is not valid
5001	PI_CNTR_PCC_SCAN_RUNNING	PicoCompensation is running, some actions can not be executed during

		scanning/recording
5002	PI_CNTR_INVALID_PCC_AXIS	Specified axis cannot be defined as PPC axis
5003	PI_CNTR_PCC_SCAN_OUT_OF_RANGE	Defined scan area is larger than the travel range
5004	PI_CNTR_PCC_TYPE_NOT_EXISTING	Specified PicoCompensation type is not defined
5005	PI_CNTR_PCC_PAM_ERROR	PicoCompensation parameter error
5006	PI_CNTR_PCC_TABLE_ARRAY_TOO_LARGE	PicoCompensation table is larger than maximum table length
5100	PI_CNTR_NEXLINE_ERROR	Common error in NEXLINE® firmware module
5101	PI_CNTR_CHANNEL_ALREADY_USED	Output channel for NEXLINE® can not be redefined for other usage
5102	PI_CNTR_NEXLINE_TABLE_TOO_SMALL	Memory for NEXLINE® signals is too small
5103	PI_CNTR_RNP_WITH_SERVO_ON	RNP can not be executed if axis is in closed loop
5104	PI_CNTR_RNP_NEEDED	Relax procedure (RNP) needed
5200	PI_CNTR_AXIS_NOT_CONFIGURED	Axis must be configured for this action
5300	PI_CNTR_FREQU_ANALYSIS_FAILED	Frequency analysis failed
5301	PI_CNTR_FREQU_ANALYSIS_RUNNING	Another frequency analysis is running
6000	PI_CNTR_SENSOR_ABS_INVALID_VALUE	Invalid preset value of absolute sensor
6001	PI_CNTR_SENSOR_ABS_WRITE_ERROR	Error while writing to sensor
6002	PI_CNTR_SENSOR_ABS_READ_ERROR	Error while reading from sensor
6003	PI_CNTR_SENSOR_ABS_CRC_ERROR	Checksum error of absolute sensor
6004	PI_CNTR_SENSOR_ABS_ERROR	General error of absolute sensor
6005	PI_CNTR_SENSOR_ABS_OVERFLOW	Overflow of absolute sensor position

**Interface errors**

0	COM_NO_ERROR	No error occurred during function call
-1	COM_ERROR	Error during com operation (could not be specified)
-2	SEND_ERROR	Error while sending data
-3	REC_ERROR	Error while receiving data
-4	NOT_CONNECTED_ERROR	Not connected (no port with specified ID open)
-5	COM_BUFFER_OVERFLOW	Buffer overflow
-6	CONNECTION_FAILED	Error while opening port
-7	COM_TIMEOUT	Timeout error
-8	COM_MULTILINE_RESPONSE	There are more lines waiting in buffer
-9	COM_INVALID_ID	There is no interface or DLL handle with the specified ID
-10	COM_NOTIFY_EVENT_ERROR	Event/message for notification could not be opened
-11	COM_NOT_IMPLEMENTED	Function not supported by this interface type
-12	COM_ECHO_ERROR	Error while sending "echoed" data
-13	COM_GPIB_EDVR	IEEE488: System error
-14	COM_GPIB_ECIC	IEEE488: Function requires GPIB board to be CIC
-15	COM_GPIB_ENOL	IEEE488: Write function detected no listeners
-16	COM_GPIB_EADR	IEEE488: Interface board not addressed correctly
-17	COM_GPIB_EARG	IEEE488: Invalid argument to function call
-18	COM_GPIB_ESAC	IEEE488: Function requires GPIB board to be SAC
-19	COM_GPIB_EABO	IEEE488: I/O operation aborted
-20	COM_GPIB_ENEB	IEEE488: Interface board not found
-21	COM_GPIB_EDMA	IEEE488: Error performing DMA
-22	COM_GPIB_EOIP	IEEE488: I/O operation started before previous

-23	COM_GPIB_ECAP	operation completed
-24	COM_GPIB_EFSO	IEEE488: No capability for intended operation
-25	COM_GPIB_EBUS	IEEE488: File system operation error
-26	COM_GPIB_ESTB	IEEE488: Command error during device call
-27	COM_GPIB_ESRQ	IEEE488: Serial poll-status byte lost
-28	COM_GPIB_ETAB	IEEE488: Return buffer full
-29	COM_GPIB_ELCK	IEEE488: Address or board locked
-30	COM_RS_INVALID_DATA_BITS	RS-232: 5 data bits with 2 stop bits is an invalid combination, as is 6, 7, or 8 data bits with 1.5 stop bits
-31	COM_ERROR_RS_SETTINGS	RS-232: Error configuring the COM port
-32	COM_INTERNAL_RESOURCES_ERROR	Error dealing with internal system resources (events, threads, ...)
-33	COM_DLL_FUNC_ERROR	A DLL or one of the required functions could not be loaded
-34	COM_FTDIUSB_INVALID_HANDLE	FTDIUSB: invalid handle
-35	COM_FTDIUSB_DEVICE_NOT_FOUND	FTDIUSB: device not found
-36	COM_FTDIUSB_DEVICE_NOT_OPENED	FTDIUSB: device not opened
-37	COM_FTDIUSB_IO_ERROR	FTDIUSB: IO error
-38	COM_FTDIUSB_INSUFFICIENT_RESOURCES	FTDIUSB: insufficient resources
-39	COM_FTDIUSB_INVALID_PARAMETER	FTDIUSB: invalid parameter
-40	COM_FTDIUSB_INVALID_BAUD_RATE	FTDIUSB: invalid baud rate
-41	COM_FTDIUSB_DEVICE_NOT_OPENED_FOR_ERASE	FTDIUSB: device not opened for erase
-42	COM_FTDIUSB_DEVICE_NOT_OPENED_FOR_WRITE	FTDIUSB: device not opened for write
-43	COM_FTDIUSB_FAILED_TO_WRITE_DEVICE	FTDIUSB: failed to write device
-44	COM_FTDIUSB EEPROM_READ FAILED	FTDIUSB: EEPROM read failed
-45	COM_FTDIUSB EEPROM_WRITE FAILED	FTDIUSB: EEPROM write failed

-46	COM_FTDIUSB_EEPROM_ERASE_FAILED	FTDIUSB: EEPROM erase failed
-47	COM_FTDIUSB_EEPROM_NOT_PRESENT	FTDIUSB: EEPROM not present
-48	COM_FTDIUSB_EEPROM_NOT_PROGRAMMED	FTDIUSB: EEPROM not programmed
-49	COM_FTDIUSB_INVALID_ARGS	FTDIUSB: invalid arguments
-50	COM_FTDIUSB_NOT_SUPPORTED	FTDIUSB: not supported
-51	COM_FTDIUSB_OTHER_ERROR	FTDIUSB: other error
-52	COM_PORT_ALREADY_OPEN	Error while opening the COM port: was already open
-53	COM_PORT_CHECKSUM_ERROR	Checksum error in received data from COM port
-54	COM_SOCKET_NOT_READY	Socket not ready, you should call the function again
-55	COM_SOCKET_PORT_IN_USE	Port is used by another socket
-56	COM_SOCKET_NOT_CONNECTED	Socket not connected (or not valid)
-57	COM_SOCKET_TERMINATED	Connection terminated (by peer)
-58	COM_SOCKET_NO_RESPONSE	Can't connect to peer
-59	COM_SOCKET_INTERRUPTED	Operation was interrupted by a nonblocked signal
-60	COM_PCI_INVALID_ID	No device with this ID is present
-61	COM_PCI_ACCESS_DENIED	Driver could not be opened (on Vista: run as administrator!)
-62	COM_SOCKET_HOST_NOT_FOUND	Host not found
-63	COM_DEVICE_CONNECTED	Device already connected

### DLL errors

-1001	PI_UNKNOWN_AXIS_IDENTIFIER	Unknown axis identifier
-1002	PI_NR_NAV_OUT_OF_RANGE	Number for NAV out of range--must be in [1..10000]
-1003	PI_INVALID_SGA	Invalid value for SGA--must be one of 1, 10, 100, 1000
-1004	PI_UNEXPECTED_RESPONSE	Controller sent unexpected response
-1005	PI_NO_MANUAL_PAD	No manual control pad installed, calls to SMA and

-1006	PI_INVALID_MANUAL_PAD_KNOB	related commands are not allowed
-1007	PI_INVALID_MANUAL_PAD_AXIS	Invalid number for manual control pad knob
-1008	PI_CONTROLLER_BUSY	Axis not currently controlled by a manual control pad
-1009	PI_THREAD_ERROR	Controller is busy with some lengthy operation (e.g., reference move, fast scan algorithm)
-1010	PI_IN_MACRO_MODE	Internal error--could not start thread
-1011	PI_NOT_IN_MACRO_MODE	Controller is (already) in macro mode--command not valid in macro mode
-1012	PI_MACRO_FILE_ERROR	Controller not in macro mode--command not valid unless macro mode active
-1013	PI_NO_MACRO_OR_EMPTY	Could not open file to write or read macro
-1014	PI_MACRO_EDITOR_ERROR	No macro with specified name on controller, or macro is empty
-1015	PI_INVALID_ARGUMENT	Internal error in macro editor
-1016	PI_AXIS_ALREADY_EXISTS	One or more arguments specified for function invalid (empty string, index out of range, ...)
-1017	PI_INVALID_AXIS_IDENTIFIER	Axis identifier is already in use by a connected stage
-1018	PI_COM_ARRAY_ERROR	Invalid axis identifier
-1019	PI_COM_ARRAY_RANGE_ERROR	Could not access array data in COM server
-1020	PI_INVALID_SPA_CMD_ID	Range of array does not fit the number of parameters
-1021	PI_NR_AVG_OUT_OF_RANGE	Invalid parameter ID specified for SPA or SPA?
-1022	PI_WAV_SAMPLES_OUT_OF_RANGE	Number for AVG out of range--must be >0
-1023	PI_WAV_FAILED	Incorrect number of samples specified for WAV
-1024	PI_MOTION_ERROR	Generation of wave failed
		Motion error: position error too large, servo is switched

-1025	PI_RUNNING_MACRO	off automatically Controller is (already) running a macro
-1026	PI_PZT_CONFIG_FAILED	Configuration of PZT stage or amplifier failed
-1027	PI_PZT_CONFIG_INVALID_PARAMS	Current settings are not valid for desired configuration
-1028	PI_UNKNOWN_CHANNEL_IDENTIFIER	Unknown channel identifier
-1029	PI_WAVE_PARAM_FILE_ERROR	Error while reading/writing wave generator parameter file
-1030	PI_UNKNOWN_WAVE_SET	Could not find description of wave form. Maybe WG.INI is missing?
-1031	PI_WAVE_EDITOR_FUNC_NOT_LOADED	The WGWaveEditor DLL function was not found at startup
-1032	PI_USER_CANCELLED	The user cancelled a dialog
-1033	PI_C844_ERROR	Error from C-844 Controller
-1034	PI_DLL_NOT_LOADED	DLL necessary to call function not loaded, or function not found in DLL
-1035	PI_PARAMETER_FILE_PROTECTED	The open parameter file is protected and cannot be edited
-1036	PI_NO_PARAMETER_FILE_OPENED	There is no parameter file open
-1037	PI_STAGE_DOES_NOT_EXIST	Selected stage does not exist
-1038	PI_PARAMETER_FILE_ALREADY_OPENED	There is already a parameter file open. Close it before opening a new file
-1039	PI_PARAMETER_FILE_OPEN_ERROR	Could not open parameter file
-1040	PI_INVALID_CONTROLLER_VERSION	The version of the connected controller is invalid
-1041	PI_PARAM_SET_ERROR	Parameter could not be set with SPA--parameter not defined for this controller!
-1042	PI_NUMBER_OF_POSSIBLE_WAVES_EXCEEDED	The maximum number of wave definitions has been exceeded
-1043	PI_NUMBER_OF_POSSIBLE_GENERATORS_EXCEEDED	The maximum number of wave generators has been exceeded
-1044	PI_NO_WAVE_FOR_AXIS_DEFINED	No wave defined for specified

		axis
-1045	PI_CANT_STOP_OR_START_WAV	Wave output to axis already stopped/started
-1046	PI_REFERENCE_ERROR	Not all axes could be referenced
-1047	PI_REQUIRED_WAVE_NOT_FOUND	Could not find parameter set required by frequency relation
-1048	PI_INVALID_SPP_CMD_ID	Command ID specified for SPP or SPP? is not valid
-1049	PI_STAGE_NAME_ISNT_UNIQUE	A stage name specified for CST is not unique
-1050	PI_FILE_TRANSFER_BEGIN_MISSING	A uuencoded file transferred did not start with "begin" followed by the proper filename
-1051	PI_FILE_TRANSFER_ERROR_TEMP_FILE	Could not create/read file on host PC
-1052	PI_FILE_TRANSFER_CRC_ERROR	Checksum error when transferring a file to/from the controller
-1053	PI_COULDNT_FIND_PISTAGES_DAT	The PiStages.dat database could not be found. This file is required to connect a stage with the CST command
-1054	PI_NO_WAVE_RUNNING	No wave being output to specified axis
-1055	PI_INVALID_PASSWORD	Invalid password
-1056	PI_OPM_COM_ERROR	Error during communication with OPM (Optical Power Meter), maybe no OPM connected
-1057	PI_WAVE_EDITOR_WRONG_PARAMNUM	WaveEditor: Error during wave creation, incorrect number of parameters
-1058	PI_WAVE_EDITOR_FREQUENCY_OUT_OF_RANGE	WaveEditor: Frequency out of range
-1059	PI_WAVE_EDITOR_WRONG_IP_VALUE	WaveEditor: Error during wave creation, incorrect index for integer parameter
-1060	PI_WAVE_EDITOR_WRONG_DP_VALUE	WaveEditor: Error during wave creation, incorrect index for floating point parameter
-1061	PI_WAVE_EDITOR_WRONG_ITEM_VALUE	WaveEditor: Error during wave creation, could not

		calculate value
-1062	PI_WAVE_EDITOR_MISSING_GRAPH_COMPONENT	WaveEditor: Graph display component not installed
-1063	PI_EXT_PROFILE_UNALLOWED_CMD	User profile mode: command is not allowed, check for required preparatory commands
-1064	PI_EXT_PROFILE_EXPECTING_MOTION_ERROR	User profile mode: first target position in user profile is too far from current position
-1065	PI_EXT_PROFILE_ACTIVE	Controller is (already) in user profile mode
-1066	PI_EXT_PROFILE_INDEX_OUT_OF_RANGE	User profile mode: block or data set index out of allowed range
-1067	PI_PROFILE_GENERATOR_NO_PROFILE	ProfileGenerator: No profile has been created yet
-1068	PI_PROFILE_GENERATOR_OUT_OF_LIMITS	ProfileGenerator: Generated profile exceeds limits of one or both axes
-1069	PI_PROFILE_GENERATOR_UNKNOWN_PARAMETER	ProfileGenerator: Unknown parameter ID in Set/Get Parameter command
-1070	PI_PROFILE_GENERATOR_PAR_OUT_OF_RANGE	ProfileGenerator: Parameter out of allowed range
-1071	PI_EXT_PROFILE_OUT_OF_MEMORY	User profile mode: out of memory
-1072	PI_EXT_PROFILE_WRONG_CLUSTER	User profile mode: cluster is not assigned to this axis
-1073	PI_UNKNOWN_CLUSTER_IDENTIFIER	Unknown cluster identifier
-1074	PI_INVALID_DEVICE_DRIVER_VERSION	The installed device driver doesn't match the required version. Please refer to the documentation to determine the required device driver version.
-1075	PI_INVALID_LIBRARY_VERSION	The library used doesn't match the required version. Please refer to the documentation to determine the required library version.
-1076	PI_INTERFACE_LOCKED	The interface is currently locked by another function. Please try again later.
-1077	PI_PARAM_DAT_FILE_INVALID_VERSION	Version of parameter DAT file

		does not match the required version. Current files are available at <a href="http://www.pi.ws">www.pi.ws</a> .
-1078	PI_CANNOT_WRITE_TO_PARAM_DATFILE	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_DATFILE	Parameter value not found in parameter DAT file.
-1086	PI_MACRO_RECORDING_NOT_ALLOWED_IN_THIS_MODE	Macro recording is not allowed in this mode of operation.
-1087	PI_USER_CANCELLED_COMMAND	Command cancelled by user input.
-1088	PI_TOO_FEW_GCS_DATA	Controller sent too few GCS data sets
-1089	PI_TOO_MANY_GCS_DATA	Controller sent too many GCS data sets
-1090	PI_GCS_DATA_READ_ERROR	Communication error while reading GCS data
-1091	PI_WRONG_NUMBER_OF_INPUT_ARGUMENTS	Wrong number of input arguments.
-1092	PI_FAILED_TO_CHANGE_CCL_LEVEL	Change of command level has failed.
-1093	PI_FAILED_TO_SWITCH_OFF_SERVO	Servo mode has failed to switch off.
-1094	PI_FAILED_TO_SET_SINGLE_PARAMETER_WHILE_PERFORMING_CST	A parameter could not be set while performing CST: CST was not performed (parameters remain unchanged).

-1095	PI_ERROR_CONTROLLER_REBOOT	Connection could not be reestablished after reboot.
-1096	PI_ERROR_AT_QHPA	Sending HPA? or receiving the response has failed.
-1097	PI_QHPA_NONCOMPLIANT_WITH_GCS	HPA? response does not comply with GCS2 syntax.
-1098	PI_FAILED_TO_READ_QSPA	Response to SPA? could not be received. Response to SPA? could not be received.
-1099	PI_PAM_FILE_WRONG_VERSION	Version of PAM file cannot be handled (too old or too new)
-1100	PI_PAM_FILE_INVALID_FORMAT	PAM file does not contain required data in PAM-file format
-1101	PI_INCOMPLETE_INFORMATION	Information does not contain all required data
-1102	PI_NO_VALUE_AVAILABLE	No value for parameter available
-1103	PI_NO_PAM_FILE_OPEN	No PAM file is open
-1104	PI_INVALID_VALUE	Invalid value
-1105	PI_UNKNOWN_PARAMETER	Unknown parameter
-1106	PI_RESPONSE_TO_QSEP_FAILED	Response to SEP? could not be received.
-1107	PI_RESPONSE_TO_QSPA_FAILED	Response to SPA? could not be received.
-1108	PI_ERROR_IN_CST_VALIDATION	Error while performing CST: One or more parameters were not set correctly.
-1109	PI_ERROR_PAM_FILE_HAS_DUPLICATE_ENTRY_WITH_DIFFERENT_VALUES	PAM file has duplicate entry with different values.
-1110	PI_ERROR_FILE_NO_SIGNATURE	File has no signature
-1111	PI_ERROR_FILE_INVALID_SIGNATURE	File has invalid signature
-10000	PI_PARAMETER_DB_INVALID_STAGE_TYPE_FORMAT	PI stage database: String containing stage type and description has invalid format.
-10001	PI_PARAMETER_DB_SYSTEM_NOT_AVAILABLE	PI stage database: Database does not contain the selected stage type for the connected controller.
-10002	PI_PARAMETER_DB_FAILED_TO_ESTABLISH_CONNECTION	PI stage database: Establishing the connection has failed.

-10003	PI_PARAMETER_DB_COMMUNICATION_ERROR	PI stage database: Communication was interrupted (e.g. because database was deleted).
-10004	PI_PARAMETER_DB_ERROR_WHILE_QUERYING_PARAMETERS	PI stage database: Querying data failed.
-10005	PI_PARAMETER_DB_SYSTEM_ALREADY_EXISTS	PI stage database: System already exists. Rename stage and try again.
-10006	PI_PARAMETER_DB_QHPA_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\_PARAMETERS\_IN\_DATABASE One or more parameter definitions are not present in stage database. Please update PISTAGES3.DB via PIUpdateFinder. Missing parameters were ignored.



# 9 Adapting Settings

## In this Chapter

Settings of the C-663.....	245
Changing Parameter Values in the C-663 .....	245
Creating or Changing a Positioner Type.....	251
Parameter Overview .....	254

### 9.1 Settings of the C-663

The properties of the C-663 and the connected positioner are stored in the C-663 as parameter values (e.g., settings for the servo algorithm (p. 28)).

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-663. The values in the nonvolatile memory are loaded to the volatile memory as default values when switching on or rebooting the C-663. 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-663

#### 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-663 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. 247) 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. 247) before you make changes in the nonvolatile memory.
- Contact our customer service department (p. 273), if the C-663 exhibits unexpected behavior.

**INFORMATION**

When you use the PC software from PI, information is loaded as parameter values from a positioner database (p. 15) into the volatile memory of the C-663.

Parameters loaded from a positioner database are marked in color in the parameter overview (p. 254).

### 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. 138).

## 9.2.2 Commands for Fast Access to Individual Parameters

The following special commands only change the corresponding parameters in the volatile memory. If necessary, the changed values must be written to the nonvolatile memory with the **WPA** command (p. 219).

### **INFORMATION**

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

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

You can find details in the command descriptions (p. 138).

## 9.2.3 Saving Parameter Values in a Text File

### **INFORMATION**

The C-663 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-663. 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-663 to specific mechanics.

### **INFORMATION**

Parameter values saved in a text file on the PC can be loaded back to the C-663 in PIMikroMove or PITerminal. The **Send file...** button is available for this purpose in the send command window. Before loading into the C-663, 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-663 and the PC (p. 63).

### 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-663: Send the **SPA?** command.
  - If you want to save the parameter values from the nonvolatile memory of the C-663: 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. 246) and the commands for quick access (p. 247).

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-663 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. 247) 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-663 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-663 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. 273).

**Requirements**

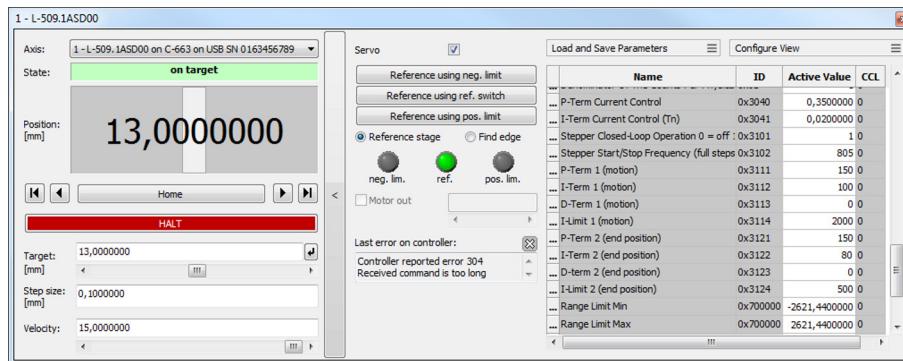
- ✓ If you want to change parameter values in the nonvolatile memory of the C-663: You have saved the parameter values of the C-663 in a text file on the PC (p. 247).
- ✓ You have established communication between the C-663 and the PC with PIMikroMove (p. 63).

**Changing parameter values: General procedure**

1. Display the parameter list in PIMikroMove.

If you want to change the axis-related parameters of the C-663:

- 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-663:

- Open the window for the system-related parameters of the C-663 in the main window of PIMikroMove by selecting **C-663 > Show system parameters** in the menu.

Name	ID	Active Value	CCL
... Ignore Macro Error?	0x72	0 0	
... Servo Update Time	0xE000200	0,000050	2

2. Change the desired parameter values in the volatile or nonvolatile memory of the C-663 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-663.
- 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-663.
- 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-663.
- 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-663. 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 parameter set appropriate 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. 15).

You can create and edit new parameter records in the PIStages3 database. This can be required in the following cases, for example:

- You want to operate a positioner with different servo control parameter settings than the one from the default parameter set.
- You want to adapt the soft limits of the positioner to your application.
- You have a custom positioner.

### INFORMATION

Possibilities for creating and editing parameter sets in the PISTAGES3.DB database:

- You can create a new positioner type easily by modifying an existing positioner type in PIMikroMove and saving it under a new name.
- You can open and edit the positioner database directly with the PIStages3Editor, which is included on the PI software CD.

PIMikroMove is used in the following for creating a new positioner type and for changing an existing positioner type.

### Requirements

- ✓ You have installed the latest version of the PISTAGES3.DB database onto your PC (p. 45).
- ✓ If PI provided a custom positioner database for your positioner, the dataset was imported into PIStages3 (p. 48).
- ✓ You have established communication with PIMikroMove between the C-663 and the PC (p. 63).

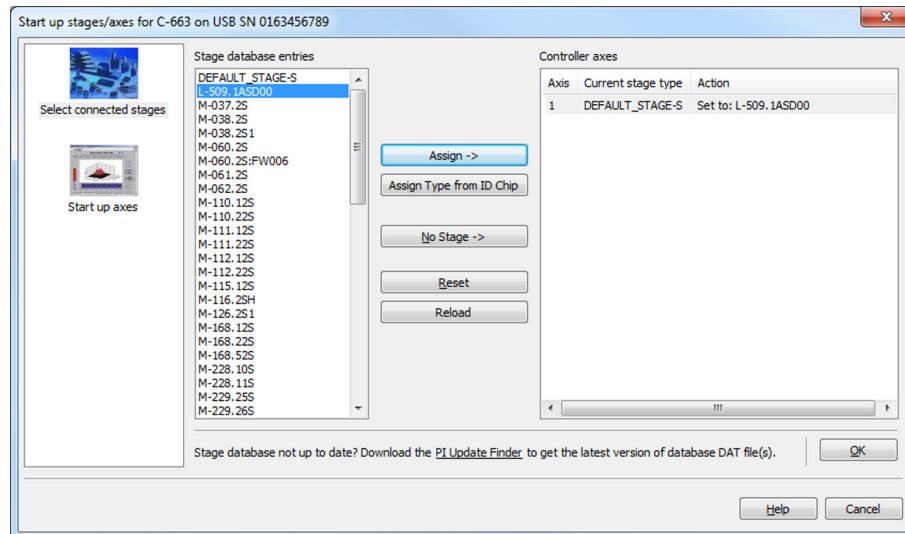
### Creating a positioner type in the positioner database

1. Select the **C-663 > Select connected stages...** menu item in the main window of PIMikroMove.

The **Start up stages/axes for C-663** window opens and the **Select connected stages** step is active.

2. Select an appropriate type of positioner during the **Select connected stages** step:
  - a) Mark the positioner type in the **Stage database entries** list.

b) Click **Assign**.

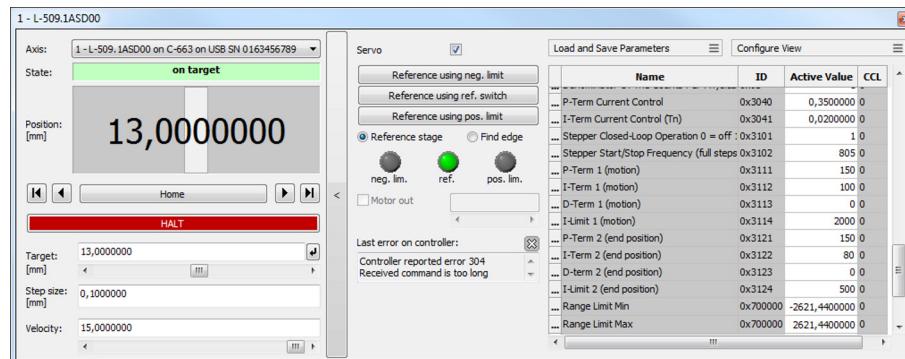


c) Confirm the selection with **OK**.

3. Click **Keep the changes temporarily** in the **Save all changes permanently** dialog to load the parameter settings into the volatile memory of the C-663.

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

4. Click **Close** in the **Start up axes** step 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.

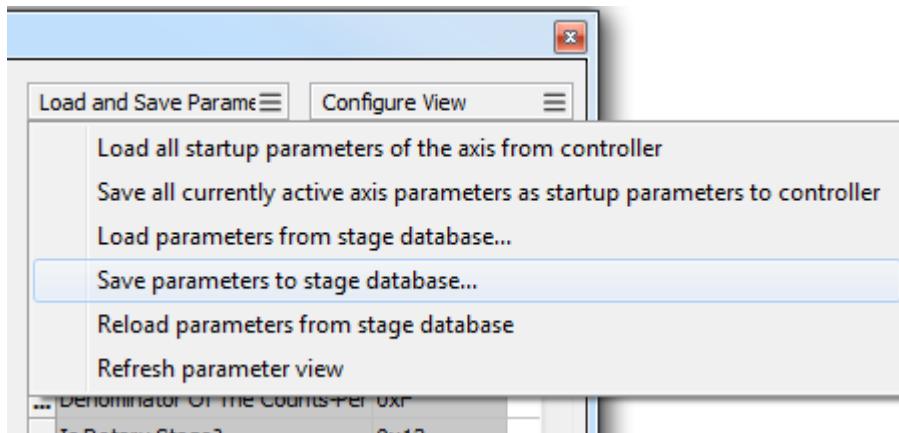


6. Enter new values for the parameters to be changed:

- If the parameter to be modified is not included in the list on the right-hand side of the window, click **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 **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 PISTAGES3.DB 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 too.

### Changing a positioner type in the positioner database

1. Select the **C-663 > Select connected stages...** menu item in the main window of PIMikroMove.  
The **Start up stages/axes for C-663** window opens, the **Select connected stages** step is active.
2. Select one of the positioners you created as described above (p. 251): 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** in the **Save Parameters as User Stage Type** 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 PISTAGES3.DB positioner database and in the main window of PIMikroMove.

## 9.4 Parameter Overview

### INFORMATION

The write access for the parameters of the C-663 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. 273).

### INFORMATION

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

Meaning of the color highlight in the parameter table:

Light gray:	The value of the parameter can be loaded from a positioner database (p. 15).
Colorless:	The value of the parameter is from one of the following sources: <ul style="list-style-type: none"> <li>▪ Factory default setting</li> <li>▪ Set by the controller during runtime and read only</li> <li>▪ Read from the ID chip of the positioner (for future use)</li> </ul>

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 that the parameter refers to, refer to "Commandable Elements" (p. 18) for further information

- Parameter name = Name of the parameter
- Description = Explanation of the parameter

<b>ID</b>	<b>Type</b>	<b>CL</b>	<b>Element</b>	<b>Parameter name</b>	<b>Description</b>
0x8	FLOAT	0	Axis	Maximum Position Error (Phys. Unit)	Maximum position error Is used for detecting motion errors. Refer to "Behavior with Motion Error" (p. 85) for details.
0xA	FLOAT	0	Axis	Maximum Closed-Loop Velocity (Phys. Unit/s)	Maximum velocity in closed-loop/open-loop operation Specifies the maximum value for parameter 0x49. Refer to "Generating a Dynamics Profile" (p. 25) for details.
0xB	FLOAT	0	Axis	Closed-Loop Acceleration (Phys. Unit/s <sup>2</sup> )	Acceleration in closed-loop/open-loop operation Limited by parameter 0x4A. Refer to "Generating a Dynamics Profile" (p. 25) for details.
0xC	FLOAT	0	Axis	Closed-Loop Deceleration (Phys. Unit/s <sup>2</sup> )	Deceleration in closed-loop/open-loop operation Limited by parameter 0x4B. Refer to "Generating a Dynamics Profile" (p. 25) for details.
0xE	INT	0	Axis	Numerator Of The Counts-Per-Physical-Unit Factor	Numerator and denominator of the factor for microsteps per physical length unit
0xF	INT	0	Axis	Denominator Of The Counts-Per-Physical-Unit Factor	For details, see "Physical Units" (p. 22).
0x13	INT	0	Axis	Is Rotary Stage?	Is this a rotation stage? 0 = Not a rotation stage 1 = Rotation stage No evaluation by the C-663, but only by the PC software: PIMikroMove determines which motion is permissible on the basis of this value.
0x14	INT	0	Axis	Has Reference?	Does the positioner have a reference switch? For details, see "Reference Switch Detection" (p. 33).

ID	Type	CL	Element	Parameter name	Description
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. 36).
0x16	FLOAT	0	Axis	Value At Reference Position (Phys. Unit)	Position value at the reference switch See examples in "Travel Range and Soft Limits" (p. 36).
0x17	FLOAT	0	Axis	Distance From Negative Limit To Reference Position (Phys. Unit)	Distance between the reference switch and the negative limit switch See examples in "Travel Range and Soft Limits" (p. 36).
0x18	INT	0	Axis	Limit Mode	Signal logic of the limit switches Refer to "Detecting Limit Switches" (p. 33) for details.
0x1A	INT	0	Axis	Has Brake?	Does the positioner have a brake? 0 = No brake present 1 = Brake present. In this case, switching the servo mode on/off and activating/deactivating the brake are coupled to each other, see BRA (p. 145) and SVO (p. 207). The brake is controlled via the <b>Motor</b> socket: <ul style="list-style-type: none"><li>▪ Pin 15, when the brake driver is integrated in the positioner</li><li>▪ Pins 9 and 16, when the C-663's integrated brake driver is used. Configuration is done with parameters 0x3094, 0x3095, 0x3096.</li></ul>
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. 36).
0x30	FLOAT	0	Axis	Maximum Travel In Negative Direction (Phys. Unit)	Soft limit in a negative direction See examples in "Travel Range and Soft Limits" (p. 36).
0x31	INT	0	Axis	Invert Reference?	Should the reference signal be inverted? For details, see "Reference Switch Detection" (p. 33).

ID	Type	CL	Element	Parameter name	Description
0x32	INT	0	Axis	Has No Limit Switches?	Does the positioner have limit switches? Refer to "Detecting Limit Switches" (p. 33) for details.
0x36	INT	0	Axis	Settling Window (encoder counts)	Settling window around the target position Refer to "On-Target State" (p. 32) for details.
0x3C	CHAR	0	Axis	Stage Name	Positioner name Maximum 20 characters, default value: DEFAULT_STAGE
0x3F	FLOAT	0	Axis	Settling Time (s)	Delay time for setting the on-target state. Refer to "On-Target State" (p. 32) for details.
0x40	INT	0	Axis	Holding Current	Holding current For details, see "Motor Control" (p. 31)
0x41	INT	0	Axis	Drive Current	Minimum value of the operating current For details, see "Motor Control" (p. 31)
0x42	INT	0	Axis	Holding Current Delay (ms)	Holding current delay time For details, see "Motor Control" (p. 31)
0x47	INT	0	Axis	Reference Travel Direction	Default direction for the reference move Refer to "Referencing" (p. 38) for details.
0x49	FLOAT	0	Axis	Closed-Loop Velocity (Phys. Unit/s)	Velocity in closed-loop/open-loop operation Limited by parameter 0xA Refer to "Generating a Dynamics Profile" (p. 25) for details.
0x4A	FLOAT	0	Axis	Maximum Closed-Loop Acceleration (Phys. Unit/s <sup>2</sup> )	Maximum acceleration in closed-loop/open-loop operation Specifies the maximum value for parameter 0xB. Refer to "Generating a Dynamics Profile" (p. 25) for details.
0x4B	FLOAT	0	Axis	Maximum Closed-Loop Deceleration (Phys. Unit/s <sup>2</sup> )	Maximum deceleration in closed-loop/open-loop operation

ID	Type	CL	Element	Parameter name	Description
					Specifies the maximum value for parameter 0xC. Refer to "Generating a Dynamics Profile" (p. 25) for details.
0x50	FLOAT	0	Axis	Velocity For Reference Moves (Phys. Unit/s)	Velocity for reference move Refer to "Referencing" (p. 38) for details.
0x5A	INT	0	Axis	Numerator Of The Servo-Loop Input Factor	Numerator and denominator of the servo-loop input factor
0x5B	INT	0	Axis	Denominator Of The Servo-Loop Input Factor	For details, see "Position Control" (p. 28).
0x5C	INT	0	Axis	Source Of Reference Signal	Reference signal source for the <code>FRF</code> or <code>FED</code> commands For details, see "Commands and Parameters for Digital Inputs" (p. 97) and "Using Digital Input Signals as Switch Signals" (p. 99).
0x5D	INT	0	Axis	Source Of Negative Limit Signal	Reference signal source for the <code>FNL</code> or <code>FED</code> commands For details, see "Commands and Parameters for Digital Inputs" (p. 97) and "Using Digital Input Signals as Switch Signals" (p. 99).
0x5E	INT	0	Axis	Source Of Positive Limit Signal	Reference signal source for the <code>FPL</code> or <code>FED</code> commands For details, see "Commands and Parameters for Digital Inputs" (p. 97) and "Using Digital Input Signals as Switch Signals" (p. 99).
0x5F	INT	0	Axis	Invert Digital Input Used For Negative Limit	Inverts the polarity of the digital inputs that are used as the source of the negative limit switch signal. For details, see "Commands and Parameters for Digital Inputs" (p. 97) and "Using Digital Input Signals as Switch Signals" (p. 99).
0x60	INT	0	Axis	Invert Digital Input Used For Positive Limit	Inverts the polarity of the digital inputs that are used as the source of the positive limit switch signal. For details, see "Commands and Parameters for Digital Inputs" (p. 97) and "Using Digital Input Signals as Switch Signals" (p. 99).

ID	Type	CL	Element	Parameter name	Description
0x61	INT	0	Axis	Invert Direction Of Motion For Joystick-Controlled Axis?	Should the direction of motion for joystick-controlled axes be inverted? Details see "Commands and Parameters for Joystick Control" (p. 103).
0x63	FLOAT	0	Axis	Distance Between Limit And Hard Stop (Phys. Unit)	Distance between the built-in limit switch and the hard stop Refer to "Referencing" (p. 38) for details.
0x70	INT	0	Axis	Reference Signal Mode	Reference signal type For details, see "Reference Switch Detection" (p. 33).
0x71	INT	0	Axis	D-Term Delay (No. Of Servo Cycles)	D term delay For details, see "Position Control" (p. 28).
0x72	INT	0	System	Ignore Macro Error?	Ignore macro error? For details, see "Commands and Parameters for Macros" (p. 111).
0x77	INT	0	Axis	Use Limit Switches Only For Reference Moves?	Should the limit switches only be used for reference moves? Refer to "Detecting Limit Switches" (p. 33) for details.
0x78	FLOAT	0	Axis	Distance From Limit To Start Of Ref. Search (Phys. Unit)	Distance between the limit switch or hard stop and the starting position for the reference move to the index pulse Refer to "Referencing" (p. 38) for details.
0x79	FLOAT	0	Axis	Distance For Reference Search (Phys. Unit)	Maximum distance for the reference move to the index pulse Refer to "Referencing" (p. 38) for details.
0x9A	INT	0	Axis	Numerator Of The Counts-Per-Physical-Unit Factor For Sensor	Numerator and denominator of the factor for sensor counts per physical length unit For details, see "Physical Units" (p. 22).
0x9B	INT	0	Axis	Denominator Of The Counts-Per-Physical-Unit Factor For Sensor	

ID	Type	CL	Element	Parameter name	Description
0x3040	FLOAT	0	Axis	P Term Current Control	Proportional constant of the current regulation For details, see "Motor Control" (p. 31).
0x3041	FLOAT	0	Axis	I Term Current Control (Tn)	Integration constant of the current regulation For details, see "Motor Control" (p. 31).
0x3094	INT	0	Axis	Internal Brake	Use the integrated brake driver of the C-663? 0 = do not use brake driver 1 = use brake driver. The brake is activated (closed) when the supply voltage drops below the value of parameter 0x3096. In addition, switching the servo mode on/off and activating/deactivating the brake are coupled to each other, see <b>BRA</b> (p. 145) and <b>SVO</b> (p. 207). The setting only takes effect when parameter 0x1A has the value 1 ("brake available"). The signals from the brake driver are output at pins 9 and 16 of the <b>Motor</b> socket.
0x3095	FLOAT	0	Axis	Brake Activation Voltage (V)	Supply voltage for releasing the brake 0 to 48 V Is only used when parameters 0x1A and 0x3094 each have the value 1.
0x3096	FLOAT	0	Axis	Brake Continuous Voltage (V)	Supply voltage for keeping the brake released permanently 0 to 48 V Should be smaller than the value of parameter 0x3095. To prevent heat from building up too much, continuous supply of voltage to the brake should be kept as low as possible. Is only used when parameters 0x1A and 0x3094 each have the value 1.
0x3101	INT	0	Axis	Stepper Closed-Loop Operation	Operating mode of the C-663 For details, see "Operating Modes" (p. 20).

ID	Type	CL	Element	Parameter name	Description
0x3102	INT	0	Axis	Stepper Start/Stop Frequency (full steps/s)	Maximum full-step frequency when starting and stopping the motion in closed-loop operation For details, see "Position Control" (p. 28).
0x3106	INT	3	Axis	Current Full-Step Frequency (full steps/s)	Current full-step frequency of the motor For details, see "Motor Control" (p. 31).
0x3111	INT	0	Axis	P Term 1 (Motion)	Proportional constant 1 For details, see "Position Control" (p. 28).
0x3112	INT	0	Axis	I Term 1 (Motion)	Integration constant 1 For details, see "Position Control" (p. 28).
0x3113	INT	0	Axis	D Term 1 (Motion)	Differential constant 1 For details, see "Position Control" (p. 28).
0x3114	INT	0	Axis	I-Limit 1 (Motion)	Limit of the integration constant 1 For details, see "Position Control" (p. 28).
0x3116	INT	0	Axis	Velocity Threshold For Switching To End-Position PID Set (Phys. Unit/s)	Velocity threshold value for switching the servo control parameters For details, see "Position Control" (p. 28).
0x3117	FLOAT	0	Axis	Kvff (motion)	Weighting of the velocity as feedforward signal For details, see "Position Control" (p. 28).
0x3121	INT	0	Axis	P Term 2 (End Position)	Proportional constant 2 For details, see "Position Control" (p. 28).
0x3122	INT	0	Axis	I Term 2 (End Position)	Integration constant 2 For details, see "Position Control" (p. 28).
0x3123	INT	0	Axis	D Term 2 (End Position)	Differential constant 2 For details, see "Position Control" (p. 28).
0x3124	INT	0	Axis	I-Limit 2 (End Position)	Limit of the integration constant 2 For details, see "Position Control" (p. 28).

ID	Type	CL	Element	Parameter name	Description
0x70000000	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. 34).
0x70000001	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. 34).
0x07000601	CHAR	0	Axis	Axis Unit	Unit symbol For details, see "Physical Units" (p. 22).
0x0D000000	CHAR	2	System	Controller S/N	Serial number of the C-663 Refer to "Type Plate" (p. 11) for details.
0xE000200	FLOAT	2	System	Servo Update Time	Servo cycle time in seconds
0xF000100	CHAR	2	Axis	Stage Type	Positioner type Format for standard positioners: x-xxx Format for customized positioners: x-xxxKxxx
0xF000200	CHAR	2	Axis	Stage Serial Number	Serial number of the positioner 9-digit number
0xF000300	CHAR	2	Axis	Stage Assembly Date	Date of manufacture of the positioner Date format: DDMMYY
0xF000400	INT	2	Axis	Stage HW Version	Version number of the positioner hardware

# 10 Maintenance

## In this Chapter

Cleaning the C-663.....	263
Updating Firmware .....	263

### 10.1 Cleaning the C-663

#### NOTICE



##### Short circuits or flashovers!

The C-663 contains electrostatic-sensitive devices that can be damaged by short-circuiting or flashovers when cleaning fluids penetrate the housing.

- Before cleaning, disconnect the C-663 from the power source by removing the mains plug.
  - Prevent cleaning fluid from penetrating the housing.
- 
- When necessary, clean the surfaces of the C-663's housing using a cloth dampened with a mild cleanser or disinfectant.

### 10.2 Updating Firmware

#### NOTICE



##### Malfunction due to faulty firmware update!

A faulty or incomplete update of the C-663's firmware may mean that the C-663 can only be made operational again by the PI customer service department.

- Update the C-663's firmware only after consulting our customer service department. If possible, ask our customer service department to do the firmware update for you.
- Before updating the firmware, make sure that you have received a suitable firmware from our customer service department and have stored it at a location that is accessible to the update program.

**INFORMATION**

The **STA** LED flashes when the C-663 is in firmware update mode. The C-663 does not leave the firmware update mode until it is **restarted** after a **successful** firmware update. If the firmware update was unsuccessful or aborted, the C-663 remains in the firmware update mode after a reboot.

If the **STA** LED still flashes, even though the C-663 has been restarted after the firmware update:

- Repeat the firmware update.
- If the update of the firmware fails, contact our customer service department (p. 273).

**INFORMATION**

If new parameters are introduced with the firmware update or the C-663 memory management is changed, an initialization of the C-663 is required after updating the firmware.

**Requirements**

- ✓ You have connected the C-663 to the PC via the USB or RS-232 interface (p. 51).
- ✓ You have made sure that the C-663 is **not** a part of a daisy chain network.
- ✓ You have made sure that **no** cable is connected to the **RS-232 Out** socket.
- ✓ The **PI Firmware Updater** program is installed on the PC (p. 45).
- ✓ 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-663 changes.
- ✓ You have saved the parameter values of the C-663 to a text file on the PC (p. 247)
- ✓ You have saved the C-663 controller macros to files on the PC (p. 120).
- ✓ You have established communication with PIMikroMove or PITerminal between the C-663 and the PC (p. 63).

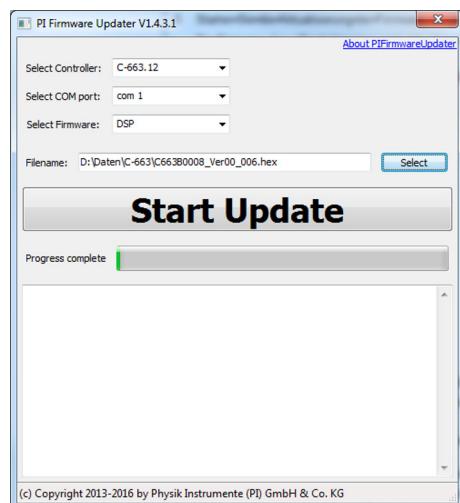
**Updating the firmware of the C-663**

1. Activate the firmware update mode in PIMikroMove or PITerminal:
  - a) If the window for sending commands is not already open in PIMikroMove, select the menu item **Tools> Command entry** in the main window or press the **F4** key on the keyboard.
  - b) Send the following commands one by one:

```
ZZZ 100 Flash
rbt
```

The C-663 reboots. If the C-663 is in firmware update mode after rebooting, the **STA** LED will flash.

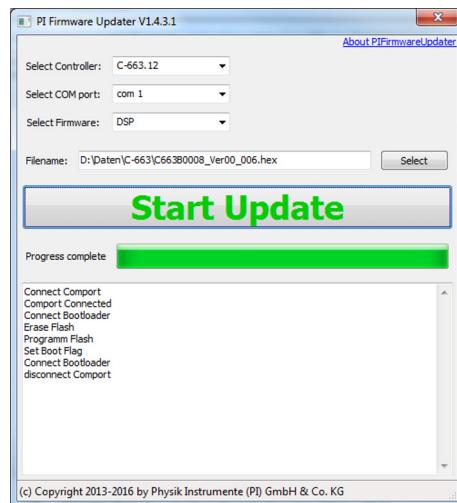
2. Close PIMikroMove or PITerminal.
3. Start the ***PI Firmware Updater*** program on the PC.  
The ***PI Firmware Updater*** window opens.
4. Set the following in the selection fields:
  - Select the *C-663.12* entry in the **Select Controller:** field.
  - Select the COM port of the PC connected to the C-663 in the **Select COM port:** field.
  - Do not change the **Select Firmware:** field. The "DSP" (DSP = Digital Signal Processor) is automatically entered by selecting the controller.
5. Select the new firmware file:
  - a) Click the **Select** button.
  - b) Go to the directory in the file selection window where you stored the firmware file.
  - c) Double-click the new firmware file (.hex extension) to enter the file path in the **Filename** field.



6. Start the firmware update by clicking on the **Start Update** button.

The firmware of the C-663 is updated. The update progress is displayed in the message list and by the progress bar.

The update was successful when the `disconnect Comport` message appears as the last entry in the message list.



7. Close the ***PI Firmware Updater*** program by clicking the cross in the top right corner of the window.
  8. Switch off the C-663 by pulling out the power cord of the power adapter from the power socket.
  9. Switch the C-663 on again by connecting the power cord of the power adapter to the power socket.
- If the firmware update was successful, the C-663 exits the firmware update mode and the **STA** LED lights up continuously.

Have new parameters been added by the firmware update, or has the memory management of the C-663 been changed?

- If no: Firmware update is finished.
- If yes: Initialization of the C-663 is required, see below.

### Initializing the C-663 after a firmware update

The initialization of the C-663 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-663 have been saved on the PC.
2. On the PC, start PIterminal or PIMikroMove, connect to the C-663, and, if necessary, open the window to send commands.
3. Initialize the C-663, 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-663.

For instructions on the general procedure, see "Changing Parameter Values: General Procedure" (p. 248).

- 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-663, see "Making Backups and Loading Controller Macros" (p. 120).



# 11 Troubleshooting

Fault: Positioner does not move	
Possible causes	Remedial measures
Cable not connected correctly	➤ Check the cable connections.
Positioner or positioner cable is defective	➤ If available, replace the defective positioner with another positioner and test the new combination.
Unsuitable positioner cable used	If unsuitable cables are used, interference can occur in the signal transmission between the positioner and the C-663. ➤ If the positioner, cable, and C-663 are marked as a related system, replace the system components with other components only after consulting PI.
Limit switch signal logic set incorrectly	In order for the positioner to be able to move, the settings of the C-663 must correspond to the limit switch logic level of the positioner; see "Limit Switch Detection" (p. 33). ➤ Adjust the <b>Limit Mode</b> parameter (0x18) accordingly.
Limit switch signals not compatible with the C-663	It is possible that positioners from third-party suppliers use unsuitable limit switch signals. ➤ Contact the customer service department (p. 273) and the manufacturer of the positioner.
Incorrect configuration	➤ Check the parameter settings of the C-663 with the commands <code>SPA?</code> (volatile memory) and <code>SEP?</code> (nonvolatile memory); for details, see "Adapting Settings" (p. 245).
Incorrect command or incorrect syntax	➤ Send the <code>ERR?</code> command and check the error code that is returned.
Wrong axis commanded	➤ Make sure that the correct axis identifier is used and that the commanded axis belongs to the correct positioner.
Joystick control is active	Motion commands are not allowed when a joystick is activated for the axis. ➤ Deactivate the joystick with the <code>JON</code> command (p. 181).

Fault: Positioner moves unintentionally	
Possible causes	Remedial measures
Joystick is not connected, but activated in the C-663	➤ Activate the joystick in the software only if the C-663 is actually connected to a joystick (p. 104).
Joystick not calibrated	➤ Calibrate the joystick (p. 105).

<b>Fault: Positioner moves unintentionally</b>	
<b>Possible causes</b>	<b>Remedial measures</b>
Startup macro is run	➤ Check whether a macro is specified as the startup macro and cancel selection of the startup macro if necessary (p. 112).
Positioner's brake deactivated with the <b>BRA</b> command (p. 145) when servo mode is switched off	➤ Secure the positioner against moving unintentionally before you deactivate the brake before you deactivate the brake by command!

<b>Fault: Positioner is oscillating or positions inaccurately</b>	
<b>Possible causes</b>	<b>Remedial measures</b>
The load was changed.	➤ Readjust the system according to the changed load (p. 76).

<b>Fault: Positioner is already oscillating during the reference move</b>	
<b>Possible causes</b>	<b>Remedial measures</b>
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 <b>Close</b> to close the <b>Start up controller</b> window instead.</li> <li>2. Open the single axis window for the connected positioner in the main window 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 <b>&gt;</b> button at the right edge of the window.</li> <li>4. Make sure the <b>Servo</b> checkbox indicates that the motor is activated.</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 deactivate the motor by unchecking the respective checkbox in the single axis window.</li> </ol>

<b>Fault: Positioner is already oscillating during the reference move</b>	
<b>Possible causes</b>	<b>Remedial measures</b>
	<p>7. Enter new values for the servo control parameters, see "Optimizing the Servo Control Parameters" (p. 76).</p> <p>8. Restart the reference move.</p> <p>9. If the positioner continues to oscillate, repeat steps 6 to 8 until the reference move has completed successfully without oscillation.</p>

<b>Fault: There is no communication between the controller and the PC</b>	
<b>Possible causes</b>	<b>Remedial measures</b>
The wrong communication cable is used or it is defective	<ul style="list-style-type: none"> <li>➤ Use a null modem cable for the RS-232 connection.</li> <li>➤ If necessary, check whether the cable works on a fault-free system.</li> </ul>
Baud rate not configured correctly	<ul style="list-style-type: none"> <li>➤ Check the settings of DIP switches 5 and 6 for the baud rate (p. 62).</li> <li>➤ In a daisy chain network make sure that the same baud rate is set for every controller.</li> </ul>
Controller address not configured correctly	<ul style="list-style-type: none"> <li>➤ Check the settings of DIP switches 1 to 4 for the controller address (p. 61).</li> </ul>
Another program is accessing the interface.	<ul style="list-style-type: none"> <li>➤ Close the other program.</li> </ul>
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 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 LF has been received.</li> </ul>

<b>Fault: The customer software does not function with the PI drivers</b>	
<b>Possible causes</b>	<b>Remedial measures</b>
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:<ul style="list-style-type: none"><li>➤ Read the information in the corresponding software manual and compare the sample code on the PI software CD with your program code.</li></ul></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. 273).

## 12 Customer Service

For inquiries and orders, contact your PI sales engineer or send us an email (<mailto:service@pi.de>).

- If you have any 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)
  - Operating system on the PC (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 (p. 4) on our website.



# 13      Technical Data

## In this Chapter

Specifications .....	275
Dimensions .....	278
System Requirements .....	279
Pin Assignment .....	279

### 13.1    Specifications

#### 13.1.1    Data Table

<b>C-663.12</b>	
Function	Mercury Step stepper motor controller
Drive types	2-phase stepper motor
Axes	1
Supported functions	Point-to-point motion. Startup macro. Data recorder for recording operating data such as velocity, position or position error. Internal safety circuitry: Watchdog timer. ID chip detection (for future use).

<b>C-663.12</b>	
Motion and control	
Controller type	PID, parameter changing during operation
Servo cycle time	50 µs
Dynamics profile	Trapezoidal velocity profile
Microstep resolution	1/2048 full step
Encoder input	A/B quadrature, TTL, RS-422; 60 MHz
Limit switches	2 × TTL, programmable
Reference switch	1 × TTL, programmable
Index switch	1 × RS-422 for index pulse
Stall detection	Automatic motor stop when a programmable position error is exceeded (only in conjunction with sensor)

<b>Electrical properties</b>	<b>C-663.12</b>
Max. output voltage*	0 V to operating voltage, for direct control of stepper motors
Max. output power	60 W
Average output power	48 W
Power consumption, full load	48 W (max.)
Power consumption without load	3 W
Current limitation per motor phase	2.5 A

<b>Interfaces and operation</b>	<b>C-663.12</b>
Communication interfaces	USB, RS-232
Motor / sensor connection	HD D-sub 26 (f)
Controller network	Up to 16 units on a single interface**
I/O lines	4 analog / digital inputs (0 to 5 V / TTL), 4 digital outputs (TTL)
Command set	PI General Command Set (GCS)
User software	PIMikroMove
Application programming interfaces	API for C / C++ / C# / VB.NET / MATLAB / Python, drivers for NI LabVIEW
Manual control	Joystick, Y-cable for 2-D motion, pushbutton box

<b>Miscellaneous</b>	<b>C-663.12</b>
Operating voltage	24 to 48 V DC from external power adapter (48 V DC power adapter in the scope of delivery)
Max. current consumption	40 mA without load (when supplied with 48 V) 80 mA without load (when supplied with 24 V)
Operating temperature range	5 to 50 °C (temperature protection switches off at excessively high temperatures)
Mass	0.48 kg
Dimensions	130 mm × 76 mm × 40 mm (incl. mounting rails)

\* Depending on the power adapter used

\*\* 16 units with USB; 6 units with RS-232

### 13.1.2 Maximum Ratings

The C-663 is designed for the following operating data:

Input on:	Maximum operating voltage	Operating frequency	Maximum current consumption
Mini-DIN 4-pin. (f)	48 V	—	3 A

Output on:	Maximum output voltage	Maximum output current	Maximum output frequency
HD D-sub 26 (f)	48 V	2.5 A	80kHz (PWM)

### 13.1.3 Ambient Conditions and Classifications

The following ambient conditions and classifications for the C-663 must be observed:

Area of application	For indoor use only
Maximum altitude	2000 m
Air pressure	1100 hPa to 0.1 hPa
Relative humidity	Highest relative humidity 80 % for temperatures up to 31 °C Decreasing linearly to 50 % relative air humidity at 40 °C
Storage temperature	0 °C to 70 °C
Transport temperature	-25 °C to +85 °C
Oversupply category	II
Protection class	I
Degree of pollution	2
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.

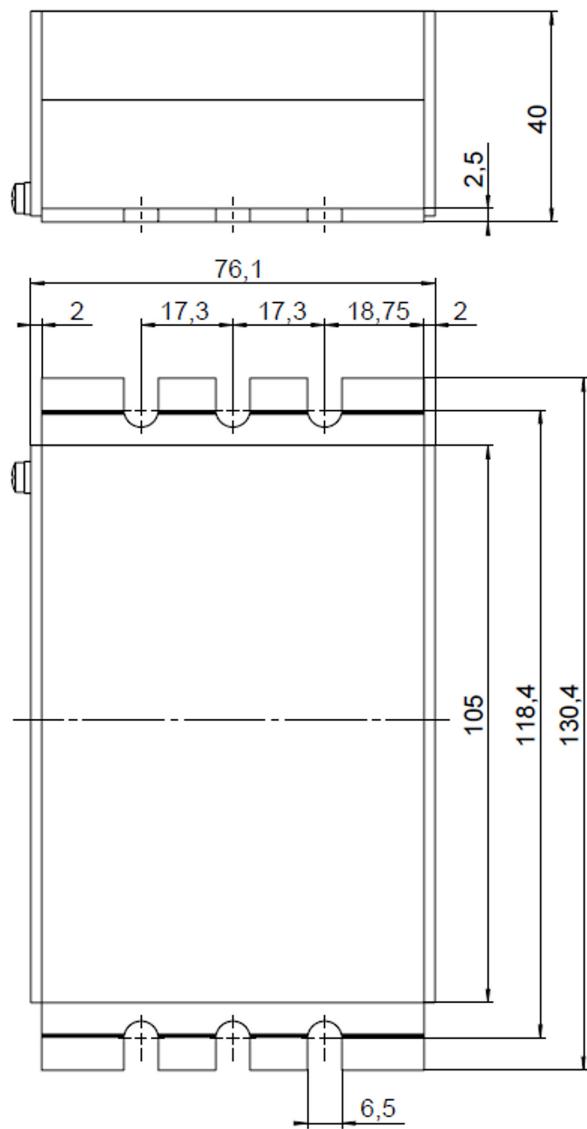


Figure 19: C-663, dimensions in mm

## 13.3 System Requirements

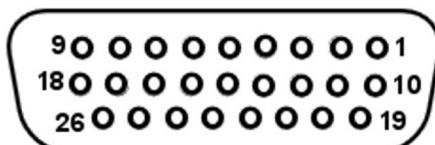
The following system requirements must be met to operate the C-663:

- C-663.12 with power adapter (included in the scope of delivery (p. 12))
- Mechanics (positioner) with stepper motor
- PC with Windows (8.1, 10; 32 bit, 64 bit) or Linux operating system and at least 30 MB free storage space
- For communication of the C-663 with the PC:
  - USB type A socket on the PC
  - USB cable type A to mini B (included in the scope of delivery)
- or
- Free COM port on the PC
- RS-232 null modem cable (included in the scope of delivery)
- Data storage medium with PC software from PI (included in the scope of delivery)

## 13.4 Pin Assignment

### 13.4.1 Motor and Sensor

**HD D-sub 26 (f)**



Pin	Signal	Direction	Function
1	OUT0a	Output	Phase I, pos. (48 V, max. 2 A, 20 kHz PWM)
2	OUT0b	Output	Phase I, pos. (48 V, max. 2 A, 20 kHz PWM)
3	OUT1a	Output	Phase I, neg. (48 V, max. 2 A, 20 kHz PWM)
4	OUT1b	Output	Phase I, neg. (48 V, max. 2 A, 20 kHz PWM)
5	OUT2a	Output	Phase II, pos. (48 V, max. 2 A, 20 kHz PWM)
6	OUT2b	Output	Phase II, pos. (48 V, max. 2 A, 20 kHz PWM)

<b>Pin</b>	<b>Signal</b>	<b>Direction</b>	<b>Function</b>
7	OUT3a	Output	Phase II, neg. (48 V, max. 2 A, 20 kHz PWM)
8	OUT3b	Output	Phase II, neg. (48 V, max. 2 A, 20 kHz PWM)
9	BRAKE_OUT	Output	Motor brake driver
10	REF	Input	Reference switch (5 V TTL input, single-ended)
11	NLIM	Input	Negative limit switch (5 V TTL input)
12	PLIM	Input	Positive limit switch (5 V TTL input)
13	-	-	Reserved
14	-	-	Reserved
15	BRAKEE	Output	Motor brake 5 V TTL, for positioners with integrated brake driver
16	VB_HC	Output	Motor brake driver (0 to 48 V supply)
17	ID Chip	Bidirectional	ID chip (intended for future use)
18	VCC_ENC	Output	Position sensor power supply (5 V, 200 mA)
19	ENCA+	Input	Encoder input A+ (RS-422)
20	ENCA-	Input	Encoder input A- (RS-422)
21	ENCB+	Input	Encoder input B+ (RS-422)
22	ENCB-	Input	Encoder input B- (RS-422)
23	INDEX+	Input	Reference switch, differential
24	INDEX-	Input	Reference switch, differential
25	GND		GND
26	VCC_ENC	Output	Position sensor power supply (5 V, 200 mA)

Do not connect anything to reserved pins.

### 13.4.2 I/O

**Mini-DIN socket, 9-pin, female**

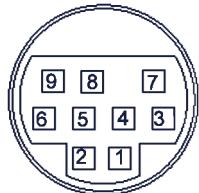


Figure 20: Front view of the mini-DIN socket

Pin	Function
1	Input 1 (analog: 0 to +5V / digital: TTL)
2	Input 2 (analog: 0 to +5V/ digital: TTL)
3	Input 3 (analog: 0 to +5V/ digital: TTL)
4	Input 4 (analog: 0 to +5V/ digital: TTL)
5	Output 1 (digital: TTL)
6	Output 2 (digital: TTL)
7	Output 3 (digital: TTL)
8	Output 4 (digital: TTL)
9	Vcc (+5 V)
Shield	GND

### 13.4.3 C-170.IO Cable for Connecting to the I/O Socket

**Mini-DIN connector, 9-pin, male, open end**

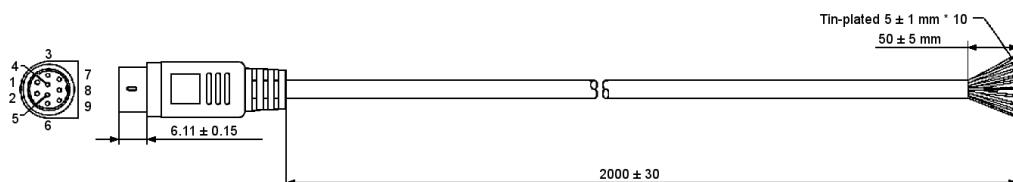


Figure 21: C-170.IO cable

<b>Pin</b>	<b>Wire Color</b>	<b>Function on the I/O socket of the C-663</b>
1	Black	Input 1 (analog: 0 to +5V / digital: TTL)
2	white	Input 2 (analog: 0 to +5V / digital: TTL)
3	Red	Input 3 (analog: 0 to +5V / digital: TTL)
4	Yellow	Input 4 (analog: 0 to +5V / digital: TTL)
5	Purple	Output 1 (digital, TTL)
6	Blue	Output 2 (digital, TTL)
7	Green	Output 3 (digital, TTL)
8	Brown	Output 4 (digital, TTL)
9	Gray	Vcc (+5V)
Sheath	Shield, coated black (thicker than the wire connected to pin 1)	GND

### 13.4.4 Joystick

**Mini-DIN socket, 6-pin, female (PS/2)**



Figure 22: Front view of Mini-DIN socket

<b>Pin</b>	<b>Function</b>
1	GND
2	Not connected
3	Output: Vcc (3.3 V)
4	Input: axis 1 of joystick 1 (0 to 3.3 V)
5	Not connected
6	Input: Button 1 of joystick 1 (0 or 3.3 V)

### 13.4.5 C-819.20Y Cable for C-819.20 Joystick

The C-819.20Y cable makes it possible to connect 2 controllers to the C-819.20 joystick.



Figure 23: Y cable C-819.20Y for joystick with 2 controllers

#### Mini-DIN connector, 6-pin, female on 2 Mini-DIN connectors, 6-pin, male

Mini-DIN 6-pin, female (to joystick)	Signal	Mini-DIN, 6-pin, male, X branch (to controller 1)	Mini-DIN, 6-pin, male, Y branch (to controller 2)
Pin 1	GND	Pin 1	Pin 1
Pin 2	Button for joystick Y axis	Not connected	Pin 6
Pin 3	Joystick power source	Pin 3	Not connected
Pin 4	Joystick X axis	Pin 4	Not connected
Pin 5	Joystick Y axis	Not connected	Pin 4
Pin 6	Button for joystick X axis	Pin 6	Not connected

### 13.4.6 RS-232 In and RS-232 Out

#### RS-232 In: D-sub 9 panel plug



**RS-232 Out: D-sub 9 socket**

<b>Pin</b>	<b>Function</b>
1	Not connected
2	RxD (PC to controller)
3	TxD (controller to PC)
4	Not connected
5	GND
6	Not connected
7	Not connected
8	Not connected
9	Not connected

**INFORMATION**

The pins of the **RS-232 In** and **RS-232 Out** sockets are connected to each other in the C-663 1:1.

**INFORMATION**

In a daisy chain network connected to the PC via the RS-232 interface of the first controller, only the PC feeds the RxD line. Depending on how performant the RS-232 driver of the PC is, the range of the network may be limited to 6 devices.

**INFORMATION**

The C-663 copies every signal that it receives from the PC via USB to the RxD line of the **RS-232 In** and **RS-232 Out** sockets. The C-663 copies the signal of the TxD line via USB to the PC.

### 13.4.7 Power Adapter Connector

**DC power socket (Kycon), 4-pole (f), lockable**

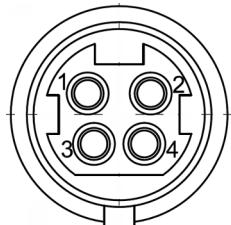


Figure 24: DC power socket (Kycon), 4-pole (f)

Pin	Signal	Direction
1	GND	GND
2	48 V DC supply voltage	Input
3	GND	GND
4	48 V DC supply voltage	Input
Shield	GND connected via the housing	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 equipment according to international, national, and local rules and regulations.

In order to fulfil its responsibility as the product manufacturer, Physik Instrumente (PI) GmbH & Co. KG undertakes environmentally correct disposal of all old PI equipment made available on the market after 13 August 2005 without charge.

Any old PI equipment can be sent free of charge to the following address:

Physik Instrumente (PI) GmbH & Co. KG  
Auf der Roemerstr. 1  
D-76228 Karlsruhe, Germany





## 15 EU Declaration of Conformity

For the C-663, an EU Declaration of Conformity has been issued in accordance with the following European directives:

EMC Directive

RoHS Directive

The applied standards certifying the conformity are listed below.

EMC: EN 61326-1

Safety: EN 61010-1

RoHS: EN 50581 or EN IEC 63000

