MS205E

# C-863 Mercury Controller

**User Manual**

Version: 2.0.0 Date: 21.03.2013

**This document describes the following product:**

- **C-863.11**
  Mercury DC Motor Controller, 1 Channel, with Wide-Range Power Supply

PIEZO NANO POSITIONING | WWW.PI.WS

Physik Instrumente (PI) GmbH & Co. KG · Auf der Römerstr. 1 76228 Karlsruhe, Germany
Telephon +49 721 4846-0 · Telefax +49 721 4846-1019 · E-Mail info@pi.ws

Original instructions
First printing: 21.03.2013
Document number: MS205E, CBo, version 2.0.0

# Contents

# 1 About this Document

## In this Chapter

## 1.1 Goal and Target Audience of this User Manual

This manual contains information on the intended use of the C-863.

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. 5) on our website.

## 1.2 Symbols and Typographic Conventions

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

***NOTICE***

**Dangerous situation**

If not avoided, the dangerous situation will result in damage to the equipment.

➢ Actions to take to avoid the situation.

***INFORMATION***

Information for easier handling, tricks, tips, etc.

| **Symbol/Label** | **Meaning** |
|---|---|
| 1.<br>2. | Action consisting of several steps whose sequential order must be observed |
| ➢ | Action consisting of one or several steps whose sequential order is irrelevant |
| ▪ | List item |
| p. 5 | Cross-reference to page 5 |
| **RS-232** | Labeling of an operating element on the product (example: socket of the RS-232 interface) |
| ⚠ | Warning sign on the product which refers to detailed information in this manual. |
| ***Start > Settings*** | Menu path in the PC software (example: to open the menu, the ***Start*** and ***Settings*** menu items must be clicked in succession) |
| `SV0?` | Command line or a command from PI's General Command Set (GCS) (example: command to get the servo mode). |
| ***Device S/N*** | Parameter name (example: parameter where the serial number is stored) |
| *5* | Value that must be entered or selected via the PC software |

## 1.3 Definition

| Term | Explanation |
|---|---|
| PC software | Software that is installed on the PC. |
| Firmware | Software that is installed on the controller. |
| Volatile memory | RAM module in which the parameters are saved when the controller is switched on (working memory). |
| Non-volatile memory | EEPROM memory chip (read-only memory), from which the default values of the parameters are loaded to the volatile memory when the controller is started. |

| Term | Explanation |
|---|---|
| Axis | Also referred to as "logical axis". The logical axis reflects the motion of the stage in the firmware of the C-863. For stages that allow motion in several directions (e. g. in X, Y and Z), each direction of motion corresponds to a logical axis. |
| Stage | Mechanical system connected to the C-863. For stages having just one motion axis the designation "axis" is synonymous with "stage". Stages that allow motion in several axes are also designated as "multiaxis stages". For these stages, a distinction must be made between the individual axes.<br><br>In this manual, actuators, i. e. drive components without a moving platform (e. g. precision linear actuators), are designated as stages as well. |
| 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. |
| Incremental position sensor | Sensor (encoder) for capturing changes of position or changes of angle. Signals from the incremental position sensor are used for axis position feedback. After switching on the controller a reference point definition must be performed before absolute target positions can be commanded and reached. |
| Control value | The control value is the input for the PWM converter of the C-863. The PWM converter converts the control value into the PWM signal for the stage. |
| Dynamics profile | Comprises the target position, velocity, and acceleration of the axis calculated by the profile generator of the C-863 for any point in time of the motion. The calculated values are called "commanded values". |
| GCS | PI General Command Set; command set for PI controllers. Piezo drivers and servo controllers can be operated conjointly with minimal programming effort thanks to the GCS. |

## 1.4 Other Applicable Documents

The devices and software tools which are mentioned in this documentation are described in their own manuals.

| Description | Document |
|---|---|
| Short version of the manual for the C-863.11 | MS205Equ User Manual Short Version |
| Mercury GCS<br>LabVIEW driver library | MS206E Software Manual |
| PI GCS 2.0 DLL for C-x63.11 | MS212E Software Manual |
| GCS array<br>data format description | SM146E Software Manual |
| PIMikroMove | SM148E Software Manual |
| PIStageEditor<br>software for the management of stage databases | SM144E Software Manual |
| PI Update Finder: Search and download updates | A000T0028 Technical Note |
| PI Update Finder: Updating PC without Internet connection | A000T0032 Technical Note |
| Adapting software that was written for C-863.10 for use with C-863.11 | A000T0029 Technical Note |

***INFORMATION***

Model C-663.11 intended for operation with stepper motors is described in a separate manual (MS208D).

## 1.5 Downloading Manuals

***INFORMATION***

If a manual is missing on our website or if there are problems in downloading:

➢ Contact our customer service department (p. 291).

The current versions of the manuals are found on our website. To download a manual, proceed as follows:

1. Open the website **http://www.pi-portal.ws**.
2. Click ***Downloads***.
3. Click the corresponding category (e. g. ***C Motion Controllers***).
4. Click the corresponding product code (e. g. ***C-863.11***).

   An overview of the available file types is shown for the selected product.
5. If ***(0 Files)*** is shown in the ***Documents*** line, log in as follows to display and download the documents:

   a) Insert the product CD in the corresponding PC drive.
   b) Open the ***Manuals*** directory.
   c) Open the Release News (e. g. ***C-663.11_Releasenews_V_x_x_x.pdf***) on the CD of the product.
   d) Find the user name and password in the **User login for software download** section in the Release News.
   e) In the ***User login*** area on the left margin in the website, enter the user name and the password in the corresponding fields.
   f) Click ***Login***.

   If ***Documents (0 Files)*** is still being displayed, no manuals are available:

   – Contact our customer service department (p. 291).
6. Click ***Documents***.
7. Click the desired manual and save it on the hard disk of your PC or on a data storage medium.

# 2 Safety

## In this Chapter

## 2.1 Intended Use

The C-863 is a laboratory device as defined by DIN EN 61010. It is intended to be used in interior spaces and in an environment which is free of dirt, oil, and lubricants.

Appropriate to its design, the C-863 is intended for the operation of PI stages equipped with DC motors or voice coil drives.

The C-863 is intended for closed-loop operation with incremental position sensors. In addition, it can read and process the reference point and limit switch signals from the stage connected.

The C-863 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-863 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-863.

- ➢ Only use the C-863 for its intended purpose, and only use it if it is in a good working order.
- ➢ Read the user manual.
- ➢ Immediately eliminate any faults and malfunctions that are likely to affect safety.

The operator is responsible for the correct installation and operation of the C-863.

- ➢ Install the C-863 near the power source so that the power plug can be quickly and easily disconnected from the mains.
- ➢ Use the supplied components (power supply, adapter and power cord (p. 16)) to connect the C-863 to the power source.

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

### 2.2.1 Organizational Measures

#### User manual

- Always keep this user manual next to the C-863.
  If the user manual is lost or damaged, contact our customer service department (p. 291).
- Add all information given by the manufacturer to the user manual, for example supplements or Technical Notes.
- If you pass the C-863 on to other users, also turn over this user manual as well as all other relevant information provided by the manufacturer.
- Only use the device on the basis of the complete user manual. If your user manual is incomplete and is therefore missing important information, property damage can result.
- Only install and operate the C-863 after having read and understood this user manual.

#### Personnel qualification

The C-863 may only be started up, operated, maintained and cleaned by authorized and qualified staff.

### 2.2.2 Safety Measures during Installation

- Install the C-863 near the power source so that the power plug can be quickly and easily disconnected from the mains.
- Only use cables and connections that meet local safety regulations.

Connecting a stage with stepper motor to a DC motor controller can cause irreparable damage.

- Only connect to the C-863 a stage with DC motor or voice coil drive.

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.

The output voltage on the **Motor +** and **Motor -** pins of the **DC Motor only** socket can be as high as the supply voltage provided by the power supply. Stages without PWM amplifier are connected to these pins and can be damaged by output voltage that is too high.

- ➢ Connect a power supply whose output voltage does **not** exceed the permissible operating voltage of the stage.

Unsuitable settings made to the servo-control parameters of the C-863 can cause the stage to oscillate. Oscillations can damage the stage and/or the load affixed to it.

- ➢ Secure the stage and all loads adequately.

### 2.2.3 Safety Measures during Start-Up

When the system settings in the non-volatile memory are changed, the original settings will be lost. Unfavorable settings can cause stage oscillation, worse settling behavior and reduced positioning accuracy.

- ➢ Only change the internal system settings for the C-863 in the non-volatile memory when necessary.
- ➢ Contact our customer service department (p. 291) if you are not sure whether a change to the system settings is necessary.

Selecting an incorrect stage type in the PC software can cause damage to the stage.

- ➢ Make sure that the stage type selected in the PC software matches the stage connected.

### 2.2.4 Safety Measures during Operation

Unsuitable settings made to the servo-control parameters of the C-863 can cause the stage to oscillate. Oscillations can damage the stage and/or the load affixed to it.

- ➢ If the stage is oscillating (unusual operating noise), immediately switch off the servo mode or disconnect the C-863 from the power source.
- ➢ Only switch on the servo mode after you have modified the servo-control parameter settings of the C-863; see "Optimizing Servo-Control Parameters" (p. 90).

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 mechanical system.

- ➢ Prevent motions in open-loop operation.

- If motions in open-loop operation are necessary:
  - Set the control value with the `SMO` command so that the axis moves with low velocity.
  - Stop the axis in time. For this purpose, use the `#24`, `STP` or `HLT` command, or set the control value to zero with the `SMO` command.
- Do **not** disable via parameter setting the evaluation of the limit switch signals by the C-863.
- Check the function of the limit switches at about 10 % to 20 % of the maximum velocity.
- In the event of a malfunction of the limit switches, stop the motion immediately.
- Ensure that the end of the travel range is approached at low velocity.
- If the moving part or the load mounted on it collides with an obstacle, switch off the motor.
- If possible, adjust the soft limits in the software used for commanding motion to your mechanical system.
- Determine the maximum velocity for your application.

If no joystick is connected to the C-863, 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-863.

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

- Secure the stage against unintentional motions before you deactivate the brake by command!

### 2.2.5 Safety Measures during Maintenance

The C-863 comprises electrostatic sensitive devices.

- Do **not** open the case of the C-863.
- Before cleaning, disconnect the C-863 from the power source by pulling the power plug.

# 3 Product Description

## In this Chapter

## 3.1 Features and Applications

The Mercury DC motor controller is the perfect solution for designing cost-efficient and flexible positioning systems in which a precision stage is to be controlled with a PC or a programmable controller. In addition to the C-863, the Mercury line comprises the successful C-663 Mercury Step stepper motor controller. The controllers of the Mercury line use the same command sets and can be networked with each other.

The product features of the C-863 include:

- RS-232 and USB interfaces
- Stand-alone operation
- Daisy chain networking for multi-axis operation
- Compatible and networkable with all Mercury line controllers, including Mercury Step
- Joystick connection for manual operation
- Non-volatile macro memory
- Parameter changing during operation
- TTL inputs for limit and reference point switches
- Control signal for motor brake
- Programmable in-/outputs

## Multi-axis operation of DC and stepper motors

The C-863.11 Mercury DC motor controller has the same command set as the C-663.11 Mercury Step stepper motor controller. Up to 16 Mercury controllers (for DC and stepper motors) can be networked and operated via the same computer interface.

Mercury networks are flexible and can also be expanded later on.

## Flexible automation

The C-863 presents a number of performance characteristics that make cost-effective implementation of automation and processing tasks in both research and industry possible. With the easily understandable programming language macros can be saved in the non-volatile memory.

A programmable start-up macro makes stand-alone operation possible: The automatic execution of internal instruction cycles is even performed without external communication upon switching-on.

Four I/O lines each are used at any one time to easily synchronize motion cycles with internal or external events. A joystick can be connected for manual control.

## User-Friendly: Comprehensive software package and two interfaces

The controller has a USB interface for easy data exchange with laptop or PC. An RS-232 interface is also available standard.

The software provided enables the networked operation of multiple controllers. LabVIEW drivers and a program library make programming system integration easier. The Mercury controllers can be actuated with the PI General Command Set (GCS) directly. With PI GCS different PI controllers, like piezo controllers and servo controllers, can be operated conjointly with minimal programming effort.

## 3.2 Model Overview

In addition to the C-863.11 DC motor controller the C-663.11 model for stages with stepper motor is also part of the Mercury controller model series.

***INFORMATION***

The hardware of the C-863.11 DC motor controller is identical to the hardware of the C-863.10 DC motor controller. The two models differ in their firmware and use different command sets. The C-863.10 and the C-863.11 cannot be networked together.

By installing the appropriate firmware a C-863.10 can be converted into a C-863.11.

➢ If you want to convert a C-863.10 into a C-863.11, contact the customer service department (p. 291).

➢ If you want to use software that you have written for the C-863.10 with the C-863.11, read A000T0029 Technical Note.

➢ For further questions contact our customer service department (p. 291).

## 3.3 Product View

### 3.3.1 Front Panel

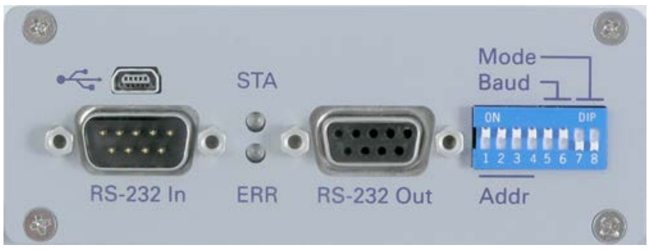


*Figure 1: C-863 Mercury DC motor controller, front view*

| Labeling | Type | Function |
|---|---|---|
| **RS-232 In** | Sub-D 9(m) (p. 301) | 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. |
| **RS-232 Out** | Sub-D 9(f) (p. 301) | Serial connection to the subsequent controller in a daisy chain network |
| ⭠ | Mini-USB type B | Universal serial bus for connecting to the PC; do not connect if **RS-232 In** is already connected. |

| Labeling | Type | Function |
|---|---|---|
| **STA** | LED<br>green/off | Controller state:<br>▪ Green: C-863 is ready for normal operation<br>▪ Off: C-863 is not connected to the supply voltage or is in firmware update mode (selection via DIP switch 8) |
| **ERR** | LED<br>red/off | Error indicator:<br>▪ Continuously lit: Error (error code ≠ 0)<br>▪ Off: No error (error code = 0)<br>The error code can be queried with the `ERR?` command. The query resets the error code to zero and the LED is switched off. |
| **Mode**, **Baud**, **Addr** | 8-bit DIP switch (p. 68) | Setting the device address, the baud rate for communication with the PC, the limit switch signal logic and the update mode. |

## 3.3.2 Rear Panel

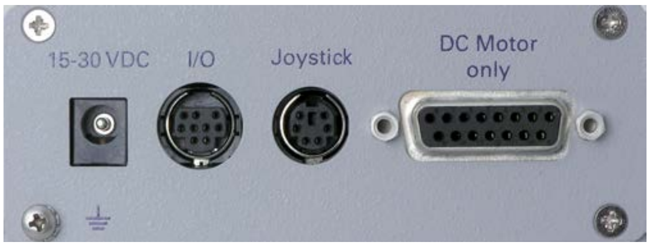


*Figure 2: C-863 Mercury DC motor controller, rear view*

| Labeling | Type | Function |
|---|---|---|
| **15-30 VDC** | Barrel connector socket (input) (p. 302) | Connection for the supply voltage |
| **I/O** | Mini-DIN socket, 9-pin (p. 298) | Digital in-/outputs:<br>▪ Outputs: Triggering external devices<br>▪ Inputs: Use in macros or as switch signals<br>Analog inputs:<br>▪ Use in macros or for scanning processes |

| Labeling | Type | Function |
|---|---|---|
| **Joystick** | Mini-DIN socket, 6-pin (p. 299) | Analog joystick<br>▪ Inputs for signals from the joystick axes and buttons<br>▪ Output for the supply voltage of the joystick |
| **DC motor only** | Sub-D 15(f) (p. 297) | Stage connection Only for DC motors!<br>▪ Output of PWM signals for the stage<br>▪ Input of the signals of the incremental position sensor<br>▪ Input of the signals from the limit switches and reference point switch |
| ⏚ | Screw and flat washer | Ground connection (p. 58)<br>If potential equalization is required, the screw can be connected to the grounding system. |

## 3.4 Scope of Delivery

| Order Number | Items |
|---|---|
| C-863.11 | Mercury DC motor controller |
| C-890.PS | Wide-Range Power Supply for Mercury Controller |
| 3763 | Power cord |
| C-815.34 | RS-232 Null-Modem Cable, 3 m, 9/9-pin |
| C-862.CN | Network Connecting Cable for Mercury, 30 cm |
| 000014651 | USB cable (type A to Mini-B) for connection to the PC |
| C-663.CD1 | Mercury product CD with software and user manuals for Mercury products<br>The Mercury product CD applies to the C-663.11 and the C-863.11. |
| MS205Dqu | Short version of the user manual for the C-863.11 |

## 3.5 Accessories

| Order Number | Description |
|---|---|
| C-815.38 | Motor Cable, 3 m, Sub-D, 15-pin (m/f) |
| C-862.CN2 | Network Connecting Cable for Mercury, 180 cm |
| C-819.20 | 2-Axis Analog Joystick for Mercury Controller, details see "Joystick Devices Available" (p. 119) |
| C-819.20Y | Y cable for connecting 2 controllers to C-819.20 joystick |
| C-819.30 | 3-Axis Analog Joystick, details see "Joystick Devices Available" (p. 120) |
| C-170.PB | Pushbutton Box, 4 Buttons and 4 LEDs.<br>Connection to the **I/O** socket of the C-863, sends 4 TTL input signals and displays the state of the 4 digital outputs via the LEDs.<br> |
| C-170.IO | I/O cable, 2 m, open end (p. 298) |

To order, contact our customer service department (p. 291).

## 3.6 Functional Principles

### 3.6.1 Block Diagram

The C-863 controls the motion of the logical axis of a stage. The following block diagram shows how the C-863 generates the output signal for the connected axis:

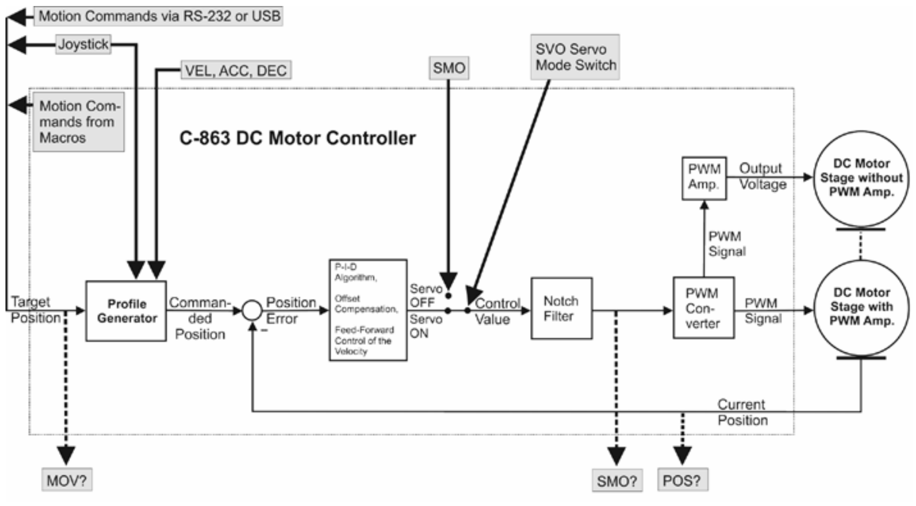


*Figure 3: C-863.11: Generation of the control value*

The C-863 supports stages with a PWM amplifier as well as stages without a PWM amplifier (p. 59).

The level of the output voltage for DC motor stages without a PWM amplifier depends on the supply voltage provided by the power supply connected to the C-863.

## 3.6.2 Commandable Items

The following table contains the items that can be accessed with commands of the GCS (p. 154).

| Item | Num-ber | Identi-fier | Description |
|---|---|---|---|
| Logical axis | 1 | 1 (modifi-able) | The logical axis represents the motion of the stage in the firmware of the C-863. It corresponds to the axis of a linear coordinate system.<br>All commands for the motion of a stage refer to logical axes.<br>Motions for logical axes are commanded in the firmware of the C-863 (i.e. for the directions of motion of a stage). The motion commands `MOV` and `MVR`, for example, are available in closed-loop operation. The motion command for open-loop operation is `SMO`.<br>The axis identifier can be queried with the `SAI?` command and modified with the `SAI` command. It can comprise up to 8 characters; valid characters are<br>1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ-_<br>The new axis identifier is saved automatically in the nonvolatile memory and is thus still available even after a reboot or after the next switching-on. |
| Analog inputs | 7 | 1 to 7 | The analog input lines with the identifiers 1 to 4 are the inputs 1 to 4 of the **I/O** socket (p. 298) number is displayed with the `TAC?` command and their values can be queried with the `TAV?` command. Note that these lines can also be used as digital inputs (see below).<br>Additional analog input lines are located on the **Joystick** socket (p. 299).<br>These lines are not output via `TAC?` and `TAV?`.<br>The values of all inputs can be recorded via record option 81 of the `DRC` command. |
| Digital outputs | 4 | 1 to 4 | 1 to 4 identify digital output lines 1 to 4 of the **I/O** socket (p. 298).<br>Further information see "Digital Output Signals" (p. 97). |
| Digital inputs | 4 | 1 to 4 | 1 to 4 identify digital input lines 1 to 4 of the **I/O** socket (p. 298), which can also be used as analog inputs (see above).<br>Further information see "Digital Input Signals" (p. 106). |

| **Item** | **Num-ber** | **Identi-fier** | **Description** |
|---|---|---|---|
| Joystick devices | 2 | 1, 2 | One axis of a joystick device can be connected to the **Joystick** socket (p. 299) of the C-863. Connection options: |
| Joystick axes per joystick device | 1 | 1 | Pin 4 (0 to 3.3 V): command as axis 1 of joystick device 1<br>or<br>Pin 2 (-10 to 10 V): command as axis 1 of joystick device 2 |
| Joystick buttons | 1 | 1 | One button of a joystick device can be connected to pin 6 of the **Joystick** socket (p. 299): command as button 1 of joystick device 1.<br>Further information see "Joystick Control" (p. 112).<br>For data recorder configuration with the `DRC` command, the following data source identifiers apply, notwithstanding the above specifications:<br>5 = axis 1 of joystick device 1<br>6 = button 1 of joystick device 1<br>7 = axis 1 of joystick device 2 |
| Data recorder table | 2 | 1, 2 | The C-863 has 2 data recorder tables (query with `TNR?`) with 1024 data points per table. |
| Controller address | 1 | 1 to 16 | The controller address can be set within the range of 1 to 16 with the DIP switches on the front panel of the C-863 (p. 69). In a daisy chain (p. 61), each controller must have a unique address (p. 146). |

## 3.6.3 Important Components of the Firmware

The firmware of the C-863 provides the following functional units:

| Firmware Component | Description |
|---|---|
| ASCII commands | Communication with the C-863 can be managed using the commands of the PI General Command Set (GCS; version 2.0). The GCS is independent of the hardware (controller, stages connected).<br>Examples of the use of GCS:<br>▪ Configuring the C-863<br>▪ Setting operating mode<br>▪ Starting stage motions<br>▪ Getting system and position values<br>You can find a list of available commands in the "Command Overview " section (p. 149). |
| Parameters | Parameters reflect the properties of the connected stage (e.g. travel range) and specify the behavior of the C-863 (e.g. settings for the servo algorithm or for using the digital inputs).<br>Parameter values can be changed to adapt the system to the particular application. Further information can be found in the "Adapting Settings" section (p. 263). |
| Profile generator and servo algorithm | In closed-loop operation, a profile generator generates the dynamics profile. The position error that results from the difference between the dynamics profile generated and the actual position (sensor feedback) runs through a PID servo algorithm. You can find further information in the sections "Generation of Dynamics Profile" (p. 24), "Servo Algorithm and Other Control Value Corrections" (p. 27) and "Motion Triggering" (p. 23). |
| Data recorder | The C-863 contains a real-time data recorder (p. 95). The data recorder can record various signals (e. g. position, analog input) from different data sources (e. g. logical axes or input channels). |
| Macros | The C-863 can save macros (p. 121). Command sequences can be defined and permanently stored in the nonvolatile memory of the device via the macro function. A start-up macro can be defined that is executed each time that the C-863 is switched on or rebooted. The start-up macro simplifies stand-alone operation (operation without a connection to the PC). Further information can be found in the "Controller Macros" section (p. 121). |

The firmware can be updated with a tool (p. 280).

### 3.6.4 Operating Modes

The C-863 supports the following operating modes:

| Operating Mode | Description |
|---|---|
| Closed-loop operation (servo mode On) | A profile generator calculates the dynamics profile from the values specified for target position, velocity, acceleration and deceleration. The position error that results from the difference between the dynamics profile generated and the actual position (sensor feedback) runs through a PID servo algorithm (**p**roportional **i**ntegral **d**ifferential). Additional corrections can be performed as well. The result is the control value for the PWM converter integrated in the C-863. You can find further information in the sections "Generation of Dynamics Profile" (p. 24) and "Servo Algorithm and Other Control Value Corrections" (p. 27). |
| Open-loop operation (servo mode Off) | In open-loop operation, the C-863 does not calculate any dynamics profile and does not evaluate the signals of the position sensor. As a result, the stage can move unbraked to the end of the travel range and, despite the limit switch function, strike the hard stop. |

***INFORMATION***

The C-863 is intended for closed-loop operation with incremental position sensors (servo mode On). After switching-on, open-loop operation is enabled by default (servo mode Off).

➢ Get the current operating mode with the `SVO?`, `#4` or `SRG?` commands.

➢ Enable closed-loop operation with the `SVO` command.

➢ If necessary, program a start-up macro that starts the C-863 via the `SVO` command in closed-loop operation; see "Setting up a start-up macro" (p. 131).

➢ Prevent motions in open-loop operation.

### 3.6.5 Physical Units

The C-863 supports various units of length for positions. The adaptation is made via a factor with which the counts of the incremental encoder are converted into the physical unit of length required. The conversion factor is set with the following parameters:

| Parameter | Description and possible values |
|---|---|
| ***Numerator Of The Counts-Per-Physical-Unit Factor***<br>0xE | Numerator and denominator of the factor for counts per physical length unit<br>1 to 1,000,000 for each parameter.<br>The factor for the counts per physical unit of length specifies the unit of length for position queries and motion commands in closed-loop operation.<br>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.<br>The factor for the counts per physical unit of length has no impact on the stability of the servo loop but is used for the input and output scaling of position values. |
| ***Denominator Of The Counts-Per-Physical-Unit Factor*** 0xF | |

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

| Parameter | Description and possible values |
|---|---|
| ***Axis Unit***<br>0x07000601 | Unit symbol<br>Maximum of 20 characters.<br>The unit symbol is "MM", for example, if the factor for the counts per physical unit of length is set with the 0xE and 0xF parameters so that the encoder counts are converted into millimeters. The unit symbol for rotation stages normally is "deg".<br>The value of the parameter 0x07000601 is not evaluated by the C-863 but is only used by the PC software for display purposes.<br>Examples:<br>1 encoder count = 100 nm<br>Counts per physical length unit: 10000:1<br>→ Unit symbol: mm<br>1 encoder count = 0.254 mm<br>Counts per physical length unit: 100:1<br>→ Unit symbol: in |

## 3.6.6 Motion Triggering

### Motions in closed-loop operation

| Trigger of the motion | Commands | Description |
|---|---|---|
| Motion commands, sent by the command line or via the PC software | MOV, MVR | Motion to absolute or relative target position |
| | GOH | Motion to zero position |
| | STE | Starts performing a step and records the step response |
| | FNL, FPL, FRF | Starting reference moves |
| | FED | Starting moves to signal edges |
| Controller macros with motion commands | MAC | Calls a macro function. Permits recording, deleting and running macros on the controller. |
| | Additional macro commands and information see "Controller Macros" (p. 121). | 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. |
| Joystick control | JON | Enables or disables a joystick device which is connected to the controller. |
| | JAX | Specifies the axis that is controlled by a joystick connected to the controller. |
| | Additional joystick commands see "Joystick Control" (p. 112). | The joystick controls the velocity of the axis (commanded velocity output from the profile generator). Motion commands are not allowed when a joystick is enabled for the axis. |

***INFORMATION***

Absolute target positions can be commanded only if the reference point definition for the axis has been performed before; see "Reference Point Definition" (p. 38).

### Motions during open-loop operation

Motions are triggered with the `SMO` command which specifies the control value for the PWM converter directly in the C-863.

Joystick control is not possible in open-loop operation.

### 3.6.7 Generation of Dynamics Profile

In closed-loop operation the profile generator performs calculations to specify the target position, velocity and acceleration of the axis for any point in time (dynamics profile). The values calculated are called commanded values. The dynamics profile generated by the profile generator of the C-863 depends on the motion parameters which are given by commands (p. 154), parameters and/or by joystick.

| Motion parameter | Com-mands | Parameters | Remarks |
|---|---|---|---|
| Acceleration (A) | `ACC` `ACC?` | Acceleration in closed-loop operation (parameter 0xB; physical unit of length/s$^2$); change with the `ACC` command or with `SPA` / `SEP`; can be saved with `WPA`. | Is limited by parameter 0x4A (maximum acceleration in closed-loop operation). |
| Deceleration (D) | `DEC` `DEC?` | Deceleration in closed-loop operation (parameter 0xC; physical unit of length/s$^2$); change with the `DEC` command or with `SPA` / `SEP`; can be saved with `WPA`. | Is limited by parameter 0x4B (maximum deceleration in closed-loop operation). |
| Velocity (V) | `VEL` `VEL?` | Velocity in closed-loop operation (parameter 0x49; physical unit of length/s); change with the `VEL` command or with `SPA` / `SEP`; can be saved with `WPA`. | Is limited by parameter 0xA (maximum velocity in closed-loop operation). When a joystick is connected to the C-863 and the joystick is enabled with the `JON` command, a factor is applied to the current velocity set with the `VEL` command. Further information see "Joystick Control" (p. 112). |
| Target position at the end of the motion | `MOV` `MVR` `GOH` `STE` | - | When a joystick is connected to the C-863 and the joystick is enabled with the `JON` command, the soft limits are set for the particular target position. When disabling the joystick device, the target position is set to the current position for the joystick-controlled axes. Further information see "Joystick Control" (p. 112). When switching on the servo mode with |

| Motion parameter | Com-mands | Parameters | Remarks |
|---|---|---|---|
| | | | 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. |

The profile generator of the C-863 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.

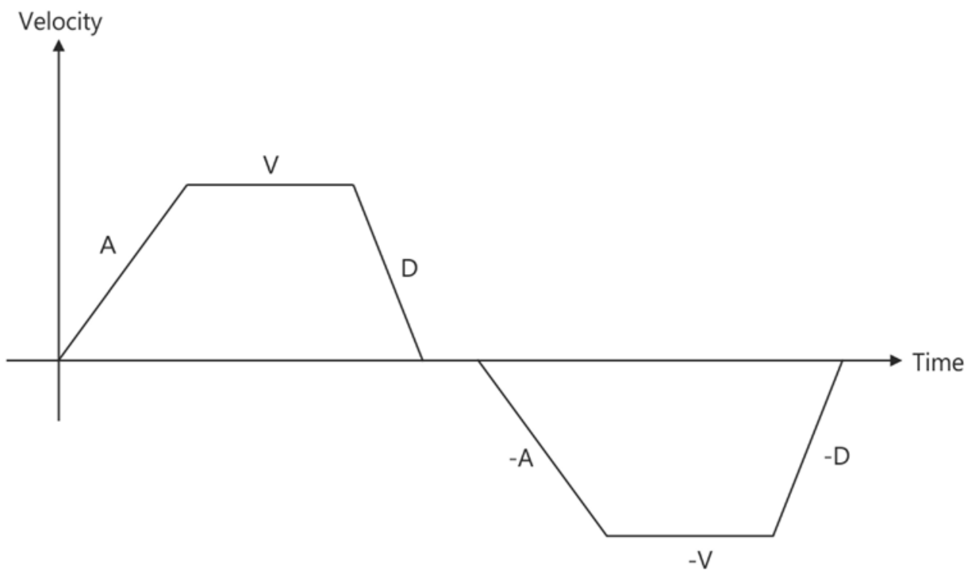


*Figure 4: Basic trapezoidal velocity profile; A = acceleration, D = deceleration, V = velocity*

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.

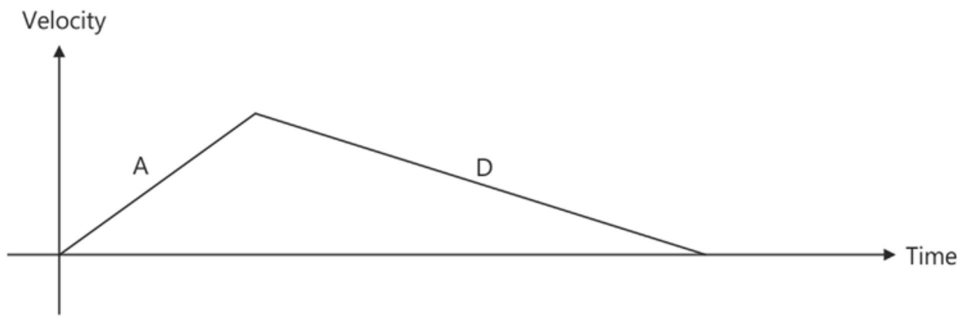


*Figure 5: Basic trapezoidal velocity profile; A = acceleration, D = deceleration, no constant velocity*

The edges for acceleration and deceleration can be symmetrical (acceleration = deceleration) or asymmetrical (acceleration ≠ 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.

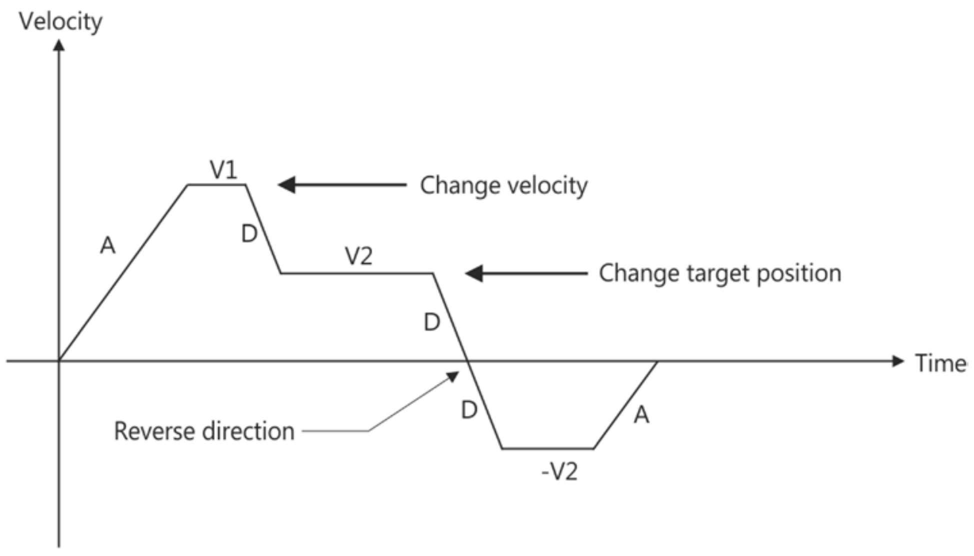


*Figure 6: Complex trapezoidal profile with parameter changes; A = acceleration; D = deceleration; V1, V2, -V2 = velocities*

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 the extent of stopping and reverse the direction of motion in order to reach the specified position.

### 3.6.8 Servo Algorithm and Other Control Value Corrections

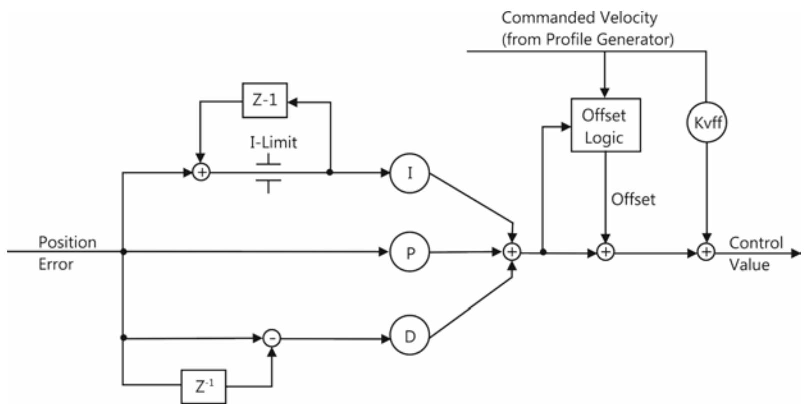


*Figure 7: PID algorithm, offset correction and feed-forward control of the velocity (KVff); the notch filter is not shown here*

In closed-loop operation, the control value for the PWM converter integrated in the C-863 and with it the settling behavior of the system are optimized via the following corrections:

- Servo algorithm: The position error, resulting from the difference between the dynamics profile generated (see "Generation of Dynamics Profile" (p. 24)) and the actual position (sensor feedback), runs through a P-I-D servo algorithm (**p**roportional **i**ntegral **d**ifferential).
- Dynamics profile corrections: The dynamics profile generated can be subjected to an offset correction and a feed-forward control of the velocity.

Independent of the operating mode, the control value can be subjected to an additional correction via the notch filter.

#### 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. 90).

| **Parameter** | **Description and possible values** |
|---|---|
| ***P-Term***<br>0x1 | Proportional constant (dimensionless)<br>0 to 32767<br>Aim: Rapid correction of the position error |

| Parameter | Description and possible values |
|---|---|
| ***I-Term***<br>0x2 | Integration constant (dimensionless)<br>0 to 32767<br>Aim: Reduction of the static position error |
| ***D-Term***<br>0x3 | Differential constant (dimensionless)<br>0 to 32767<br>Aim: Damping of rapid control oscillations |
| ***I-Limit***<br>0x4 | Limit of the integration constant (dimensionless)<br>0 to 32767 |
| ***D-Term Delay (No. Of Servo Cycles)***<br>0x71 | D-term delay<br>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. |

***INFORMATION***

For reason of compatibility the C-863 has the following additional set of servo-control parameters:

- Parameters 0x411, 0x412, 0x413, 0x414

The values of these parameters are automatically set to the actually used values of the servo-control parameters 0x1, 0x2, 0x3 and 0x4.

➢ Only use parameters 0x1, 0x2, 0x3 and 0x4 for optimizing the settling behavior of the system; see "Optimizing Servo-Control Parameters" (p. 90).

The input of the servo algorithm can be configured for the C-863 with the following parameters:

| Parameter | Description and possible values |
|---|---|
| ***Numerator Of The Servo-Loop Input Factor***<br>0x5A | Numerator and denominator of the servo loop input factor<br>1 to 1,000,000 for both parameters<br>The servo loop input factor decouples the servo-control parameters from the encoder resolution.<br>The servo loop input factor is independent of the factor for counts per physical length unit (0xE and 0xF).<br>Numerator and denominator of the servo loop input factor should not be changed. |
| ***Denominator Of The Servo-Loop Input Factor***<br>0x5B | |

### Corrections of the dynamics profile

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

| **Parameter** | **Description and possible values** |
|---|---|
| ***Motor Offset Positive***<br>0x33 | Offset for the positive direction of motion (dimensionless).<br>0 to 32766 |
| ***Motor Offset Negative***<br>0x34 | Offset for the negative direction of motion (dimensionless).<br>0 to 32766 |
| ***Motor Drive Offset***<br>0x48 | Velocity-dependent offset (dimensionless). Is used if the commanded velocity does not equal zero (i.e. if the end of the dynamics profile has still not been reached).<br>0 to 32766 |
| ***Kvff***<br>0x5 | Feed-forward control of the commanded velocity<br>0 to 32767<br>Aim: Minimization of the position error |

### Control value corrections regardless of the operating mode

The parameters listed below correct the control value both in closed- and open-loop operation.

| **Parameter** | **Description and possible values** |
|---|---|
| ***Notch Filter Frequency 1 (Hz)***<br>0x94 | Frequency of the first notch filter<br>40 to 20000 Hz<br>The appropriate frequency component is reduced in the control value to compensate for undesired resonances in the mechanical system. An adjustment can be particularly useful in the case of very high loads. |
| ***Notch Filter Edge 1***<br>0x95 | Rise of the edge of the first notch filter (dimensionless)<br>0.1 to 10<br>This parameter value should not be changed. |

### 3.6.9 On-Target State

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

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

The C-863 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:

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

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

In the *On Target* trigger mode (p. 101) the on-target state of the selected axis is output at the selected trigger output.

| Parameter | Description and possible values |
|---|---|
| ***Settling Time(s)***<br>0x3F | Delay time for setting the on-target state<br>0 to 1.000 s |
| ***Settling Window (encoder counts)***<br>0x36 | Settling window around the target position<br>0 to $2^{31}$ counts of the incremental encoder<br>Specifies the window limits. If the current position exits the settling window, the target position is no longer considered as reached.<br>The parameter value corresponds to half the width of the window. It can be changed only if the servo mode is switched off. |

### 3.6.10 Reference Point Switch Detection

The C-863 receives the signal of one reference point switch on pin 13 of the **DC Motor only** socket. (p. 297)

Reference point switch detection can be configured by the C-863 with the following parameters:

| Parameter | Description and possible values |
|---|---|
| ***Invert Reference?***<br>0x31 | Should the reference signal be inverted?<br>0 = Reference signal not inverted<br>1 = Reference signal inverted<br>This parameter is used for inverting the reference signal whose source can be either the reference point switch or a digital input which is used instead of the reference switch (p. 109). |
| ***Has Reference?***<br>0x14 | Does the stage have a reference point switch?<br>0 = No reference point switch installed<br>1 = Reference point switch present (signal input on Sub-D 15 (f) motor connection)<br>This parameter enables or disables reference moves to the reference point switch installed. |
| ***Reference Signal Type***<br>0x70 | Reference signal type<br>0 = Direction-sensing reference point switch (default setting). The signal level changes when passing the reference point switch.<br>1 = Pulse signal with a pulse width of several nanoseconds (parameter 0x47 must be set correctly). |

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

### 3.6.11 Limit Switch Detection

The C-863 receives limit switch signals at the **DC Motor only** socket (p. 297):

- Pin 5: positive limit switch
- Pin 12: negative limit switch

Reference point switch detection can be configured by the C-863 with the following parameters:

| Parameter | Description and possible values |
|---|---|
| ***Limit Mode***<br>0x18 | Signal logic of the limit switches<br>0 = Positive limit switch active high (pos-HI), negative limit switch active high (neg-HI)<br>1 = Positive limit switch active low (pos-LO), neg-HI<br>2 = pos-HI, neg-LO<br>3 = pos-LO, neg-LO<br>The C-863 can also be adapted via DIP switch 7 to the logic level of the limit switches of the stage connected (active high or active low). So that the stage can move, the DIP switch **and** parameter settings must correspond to the limit switch logic level (p. 70). |
| ***Has No Limit Switches?***<br>0x32 | Does the stage have limit switches?<br>0 = Stage has limit switches (signal inputs on Sub-D 15 (f) motor connection)<br>1 = Stage has no limit switches<br>This parameter enables or disables a stop of the motion at the limit switches installed. |

The signals from the limit switches (also end-of-travel sensors) of a linear positioning stage are used to stop the motion prior to the hard stop at both ends of the travel range. Because the set deceleration is not taken into account here, there is a risk that at high velocities the stage will strike the hard stop anyway. To prevent this, soft limits (p. 33) can be set via parameters of the C-863.

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 "Reference Point Definition" (p. 38).

### 3.6.12 Travel Range and Soft Limits

The following parameters of the C-863 reflect the physical travel range of the stage and define soft limits:

| Parameter | Description and possible values |
|---|---|
| ***Maximum Travel In Positive Direction (Phys. Unit)*** 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. |
| ***Value At Reference Position (Phys. Unit)*** 0x16 | Position value at the reference point switch (physical unit) The current position is set to this value if the axis has executed a reference move to the reference point switch (start with `FRF`). The parameter value is used in addition for calculating the position values which are set after reference moves to the limit switches; this also applies when the mechanical system does not have a reference point switch. |
| ***Distance From Negative Limit To Reference Position (Phys. Unit)*** 0x17 | Distance between reference point switch and negative limit switch (physical unit) If the axis has executed a reference move to the negative limit switch (start with `FNL`), the current position is set to the difference between the values of parameters 0x16 and 0x17. |
| ***Distance From Reference Position To Positive Limit (Phys. Unit)*** 0x2F | Distance between reference point switch and positive limit switch (physical unit) If the axis has executed a reference move to the positive limit switch (start with `FPL`), the current position is set to the sum of the values of parameters 0x16 and 0x2F. |
| ***Maximum Travel In Negative Direction (Phys. Unit)*** 0x30 | Soft limit in negative direction (physical unit) Based on the zero position. If this value is larger than the position value for the negative limit switch (which results from the difference of the parameters 0x16 and 0x17), the negative limit switch cannot be used for reference moves. The value can be negative. |

| Parameter | Description and possible values |
|---|---|
| ***Range Limit Min***<br>0x07000000 | Additional soft limit for the negative direction of motion (physical unit)<br>If the current position reaches this value in either open-loop or closed-loop operation, the control value is set to zero and the motion is stopped as a result. The axis can move again as soon as the value for the soft limit has been decreased. |
| ***Range Limit Max***<br>0x07000001 | Additional soft limit for the positive direction of motion (physical unit)<br>If the current position reaches this value in either open-loop or closed-loop operation, the control value is set to zero and the motion is stopped as a result. The axis can move again as soon as the value for the soft limit has been increased. |

***INFORMATION***

The C-863 supports two parameter pairs for establishing soft limits. They are intended for different applications:

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

## Examples

The following examples refer to a stage with incremental sensor, reference point switch and limit switches.

The distance between the negative and positive limit switches of the stage is 20 mm. The reference point 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 stage is reflected in the following parameters:

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

***INFORMATION***

The switch setup of the stage 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 point 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, during reference moves, specifies the position value for the reference point switch and is included in the calculation of the position values for the limit switches, 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

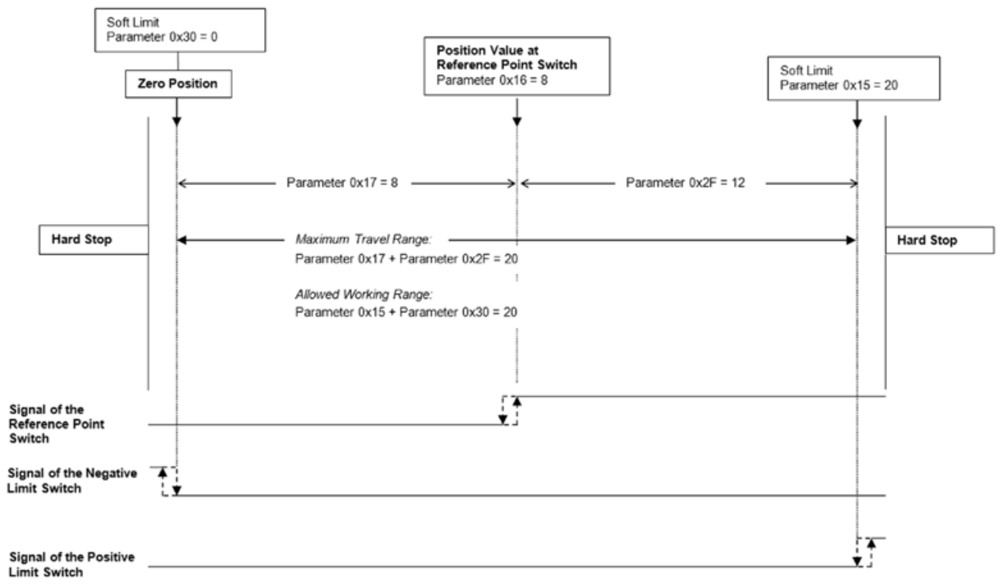

Soft Limit
Parameter 0x30 = 0
Zero Position
Position Value at Reference Point Switch
Parameter 0x16 = 8
Soft Limit
Parameter 0x15 = 20
Parameter 0x17 = 8
Parameter 0x2F = 12
Hard Stop
Maximum Travel Range:
Parameter 0x17 + Parameter 0x2F = 20
Hard Stop
Allowed Working Range:
Parameter 0x15 + Parameter 0x30 = 20
Signal of the Reference Point Switch
Signal of the Negative Limit Switch
Signal of the Positive Limit Switch

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

After a reference move of the stage to the reference point switch (`FRF` command), query commands return the following responses:

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

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

The zero position should be located at approximately a third of the distance between the negative limit switch and the reference point 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

As a result, the stage can move from the zero position 16.4 mm in the positive direction and 2.1 mm in the negative direction, respectively. The limit switches can no longer be used for reference moves.

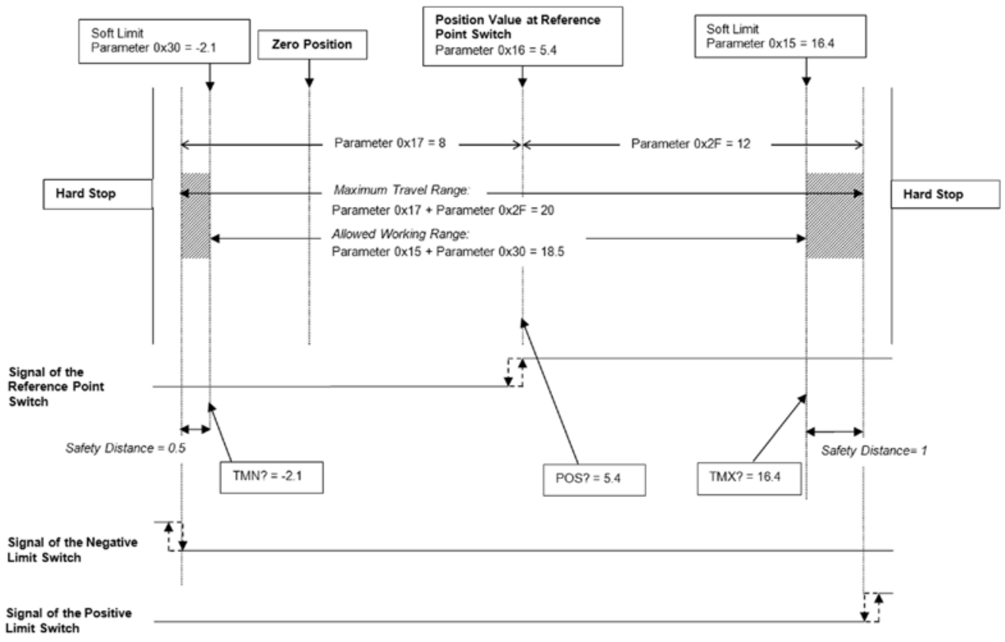


*Figure 9: The travel range of the stage is limited by soft limits.*

After a reference move of the stage to the reference point 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

### 3.6.13 Reference Point Definition

The incremental sensors that are used for axis position feedback only return relative motion information. As a result, the controller does not know the absolute position of an axis upon switching-on. So that absolute target positions can be commanded and reached, a reference point definition must be performed beforehand.

The reference point definition can be performed in different ways:

- **Reference move** (default): A reference move moves the axis to a point firmly defined, e.g. to the reference point 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.
- **Manually setting the absolute position**: If this type of reference point definition was enabled with the `RON` (p. 217) command, you can set the current position of the axis to an arbitrary point at an arbitrary value using the `POS` (p. 215) command. Here the axis is not moved. The controller knows the absolute axis position afterwards.

***INFORMATION***

During start-up using PIMikroMove, the reference point definition is performed via a reference move by default. Knowledge of the commands and parameters described here is not needed for reference point definition using PIMikroMove.

***INFORMATION***

To achieve maximum repeatability of the reference point definition, every reference move is comprised of the following steps:

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

The lower the velocity is upon reaching the switch, the less the overshoot will be and the higher the repeatability. Therefore the value of parameter 0x50 should at a maximum be as great 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$^2$)***)

## Commands

The following commands are available for reference point definition:

| Command | Syntax | Function |
|---|---|---|
| `RON` | RON {<AxisID> <ReferenceOn>} | Sets mode of reference point definition:<br>▪ <ReferenceOn> = 0: For reference point definition of the axis, an absolute position value can be assigned with POS or a reference move can be started with FRF, FNL or FPL.<br>▪ <ReferenceOn> = 1 (default): For reference point definition of the axis, a reference move must be started with FRF, FNL or FPL. Using POS is not permitted. |
| `RON?` | RON? [{<AxisID>}] | Gets mode of reference point definition. |
| `FRF` | FRF [{<AxisID>}] | Starts a reference move to the reference point switch. The approach is always made from the same side irrespectively of the axis position when sending the command. |
| `FRF?` | FRF? [{<AxisID>}] | Queries whether the reference point for an axis has already been defined.<br>1 = Reference point has been defined<br>0 = Reference point has not been defined |
| `FNL` | FNL [{<AxisID>}] | Starts a reference move to the negative limit switch. |

| Command | Syntax | Function |
|---|---|---|
| FPL | FPL [{<AxisID>}] | Starts a reference move to the positive limit switch. |
| POS | POS {<AxisID> <Position>} | Sets the current position (does not trigger a motion) and thus defines the reference point. |

## Parameters

Reference moves can be configured with the following parameters:

| Parameters | Description and possible values |
|---|---|
| ***Closed-Loop Deceleration (Phys. Unit/s$^2$)***<br>0xC | Deceleration in closed-loop operation<br>Details see "Generation of Dynamics Profile" (p. 24). |
| ***Reference Travel Direction***<br>0x47 | Default direction for the reference move<br>0 = automatic detection<br>1 = negative direction<br>2 = positive direction |
| ***Closed-Loop Velocity (Phys. Unit/s)***<br>0x49 | Velocity in closed-loop operation<br>Details see "Generation of Dynamics Profile" (p. 24). |
| ***Velocity For Reference Moves (Phys. Unit/s)***<br>0x50 | Velocity for reference move<br>Specifies the maximum velocity during a reference move for the second approach of the switch selected. For high repeatability during a reference point definition, this value should at a maximum be as great as the value of parameter 0x49. If the value of parameter 0x50 is set to 0, reference moves are not possible. |
| ***Distance Between Limit And Hard Stop (Phys. Unit)***<br>0x63 | Distance between internal limit switch and hard stop<br>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). |

***INFORMATION***

➢ For maximum repeatability always execute the reference move 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. 33).

***INFORMATION***

For reference moves, you can also use the digital inputs of the C-863 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. 109) for more information.

***INFORMATION***

If the absolute position of the axis is defined manually, 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 reference point definition is not possible otherwise.

## 3.7 Communication Interfaces

### Communication interfaces available

The C-863 can be controlled with ASCII commands from a PC: Connecting to the PC can be effected via a direct connection or via a daisy chain network. The following interfaces of the C-863 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 | 38400<br>Settings for DIP switches 5 and 6; see "Baud Rate" (p. 70)<br>Other:<br>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" (p. 2).

## 3.8 Overview of PC Software

The following table shows the PC software that is included in the product CD. The given operating systems stand for the following versions:

- Windows: versions XP, Vista and 7
- Linux: Kernel 2.6, GTK 2.0, glibc 2.4

| PC software | Operating system | Short description | Recommended use |
|---|---|---|---|
| Dynamic program library for GCS | Windows, Linux (USB only in Windows) | Allows software programming for the C-863 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.<br>Is required for PIMikroMove.<br>Is required for LabVIEW drivers if communication is to be established via USB or a daisy chain network. |
| LabVIEW drivers | Windows, Linux | LabVIEW is a software for data acquisition and process control (must be ordered separately from National Instruments). The C-863 LabVIEW software is a collection of virtual instrument drivers (VI drivers) for the C-863 controller. These drivers support the GCS. | For users who want to use LabVIEW to program their application. |

| PC software | Operating system | Short description | Recommended use |
|---|---|---|---|
| PIMikroMove | Windows | Graphic user interface for Windows with which the C-863 and other controllers from PI can be used.<br>■ The system can be started without programming effort<br>■ Graph of motions in open-loop and closed-loop operation<br>■ Macro functionality for storing command sequences on the PC (host macros)<br>■ Joystick support<br>■ Complete environment for command entry, for trying out different commands<br>No command knowledge is necessary to operate PIMikroMove. PIMikroMove uses the dynamic program library to supply commands to the controller. | For users who want to perform simple automation tasks or test their equipment before or instead of programming an application. A log window showing the commands sent makes it possible to learn how to use the commands. |
| PITerminal | Windows | Basic graphic user interface for Windows, which can be used for nearly all PI controllers (see the description of the ***Command Entry*** window in the PIMikroMove user manual). | For users who want to send GCS commands directly to the controller. |
| PIStageEditor | Windows | Program for opening and editing stage databases. | For users who want to deal intensively with the contents of stage databases. |
| PI Update Finder | Windows | Checks the PI software installed on the PC. If more current versions of the PC software are available on the PI server, downloading is offered. | For users who want to update the PC software. |
| TMS320F28xx Updater | Windows | Program for user support when updating firmware. | 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. |

## 3.9 Stage Databases

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

| Database file name | Editable? | Description |
|---|---|---|
| PIStages2.dat | No, updates can be downloaded from the PI website (p. 54). | Includes parameter sets for all standard stages from PI; is automatically saved to the PC during the installation of the software. |
| PI_UserStages2.dat | Yes, new parameter sets can be created, edited and saved (p. 268). | Is created when you make a connection to your stage using the PC software for the first time (i. e. when selecting the stage in PIMikroMove or when using the commands VST? or CST). |
| M-xxx.dat | No, you receive updates from our customer service department (p. 291). | Includes the parameter set for a custom stage, for installation see "Installing a Custom Stage Database" (p. 56). |

You can find further information in the user manuals for PIMikroMove, PIStageEditor and the PI GCS program library.

# 4 Unpacking

1. Unpack the C-863 with care.
2. Compare the contents against the items covered by the contract and against the packing list.
3. Inspect the contents for signs of damage. If parts are missing or you notice signs of damage, contact PI immediately.
4. Keep all packaging materials in case the product needs to be returned.

# 5 Quick Start

*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.

*NOTICE*

**Oscillations!**

Unsuitable settings made to the servo-control parameters of the C-863 can cause the stage to oscillate. Oscillations can damage the stage and/or the load affixed to it.

➢ Secure the stage and all loads adequately.

➢ If the stage is oscillating (unusual operating noise), immediately switch off the servo mode or disconnect the C-863 from the power source.

➢ Only switch on the servo mode after you have modified the servo-control parameter settings of the C-863; see "Optimizing Servo-Control Parameters" (p. 90).

➢ If, due to a very high load, oscillations occur already during the reference move, follow the instructions for the reference move in "Troubleshooting" (p. 285).

The aim of quick start is to start in the PIMikroMove PC software initial test motions of a stage that is connected to a non-networked C-863.

1. Install the following on the PC:
   - the PC software and the USB drivers from the product CD
   - updates for PC software and PIStages2.dat stage database
   - if provided separately by PI: custom stage database(s)

   Details see "Installing the PC Software" (p. 53).
2. Connect the C-863 via screw designated with the ground connection symbol to the grounding system (p. 58).

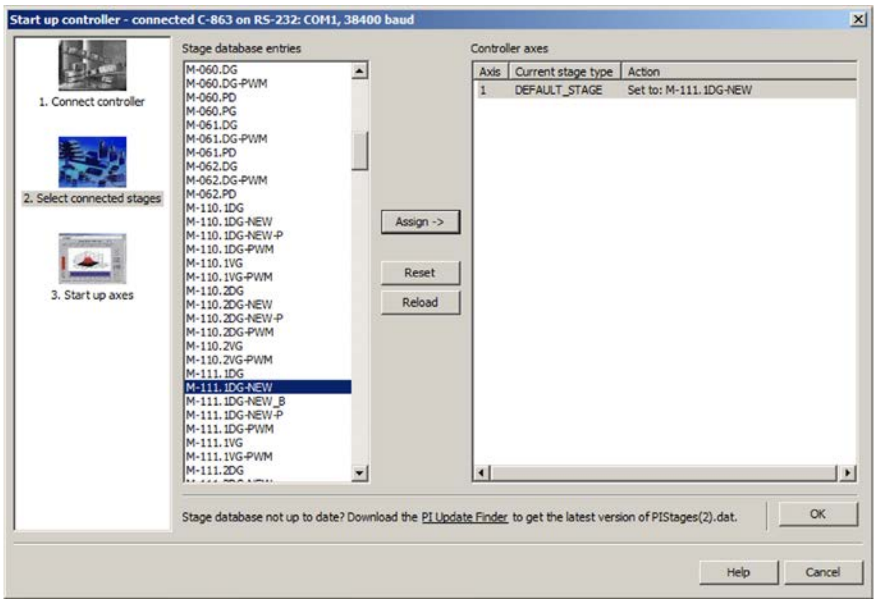
3. Connect the following to the C-863:
   - the included wide-range-input power supply (**not** connected to the wall socket via the power cord) to the **15-30 VDC** connection. Details see "Connecting Power Supply".
   - the stage to the **DC Motor only** socket. Details see "Connecting Stage" (p. 59).
   - the PC via the RS-232 interface (**RS-232 In** socket) **or** via the USB interface. Details see "Connecting PC" (p. 60).
4. Check the DIP switch settings (p. 68) and adapt them to your application if necessary. Controller address 1 must be set.
5. Switch on the C-863 (p. 71) by connecting the power cord of the wide-range-input power supply to the power socket.
6. Start PIMikroMove on the PC.
7. Establish communication between the C-863 and the PC in PIMikroMove via the RS-232 interface or the USB interface. Details see "Establishing Communication" (p. 72).
8. If in PIMikroMove the ***Select connected stages*** step is displayed, select the stage type of the stage connected:

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

   - Click on ***OK***.

   If the listed stage type is not correct:

   a) Mark the stage type in the ***Stage database entries*** list.
   b) Click ***Assign***.
   c) Confirm the selection with ***OK*** to load the parameter settings for the selected stage type from the stage database to the volatile memory of the C-863.


Start up controller - connected C-863 on RS-232: COM1, 38400 baud

Stage database entries

```
M-060.DG
M-060.DG-PWM
M-060.PD
M-060.PG
M-061.DG
M-061.DG-PWM
M-061.PD
M-062.DG
M-062.DG-PWM
M-062.PD
M-110.1DG
M-110.1DG-NEW
M-110.1DG-NEW-P
M-110.1DG-PWM
M-110.1VG
M-110.1VG-PWM
M-110.2DG
M-110.2DG-NEW
M-110.2DG-NEW-P
M-110.2DG-PWM
M-110.2VG
M-110.2VG-PWM
M-111.1DG
M-111.1DG-NEW
M-111.1DG-NEW_B
M-111.1DG-NEW-P
M-111.1DG-PWM
M-111.1VG
M-111.1VG-PWM
M-111.2DG
```

Controller axes

| Axis | Current stage type | Action |
|---|---|---|
| 1 | DEFAULT_STAGE | Set to: M-111.1DG-NEW |

Stage database not up to date? Download the PI Update Finder to get the latest version of PIStages(2).dat.

After clicking ***OK*** the ***Start up controller*** window goes to the ***Start up axes*** step.

9. In the ***Start up axes*** step, execute the reference move for the axis so that the controller knows the absolute axis position:

   – If you want to start the reference move to the reference point switch, click on ***Ref. switch***.

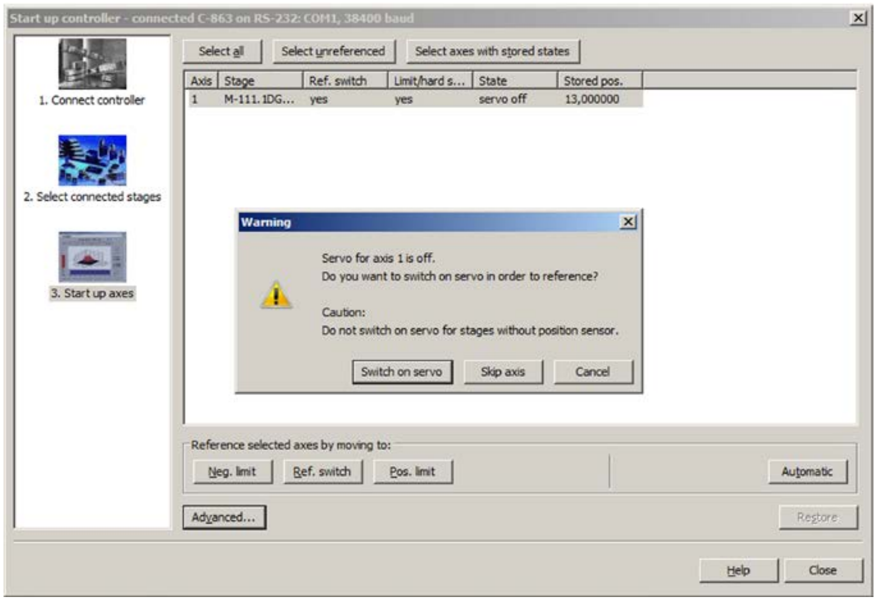
   – If you want to start the reference move to the negative limit switch, click on ***Neg. limit***.

   – If you want to start the reference move to the positive limit switch, click on ***Pos. limit***.

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

   – Switch on the servo mode by clicking on the ***Switch on servo*** button.

The axis executes the reference move.



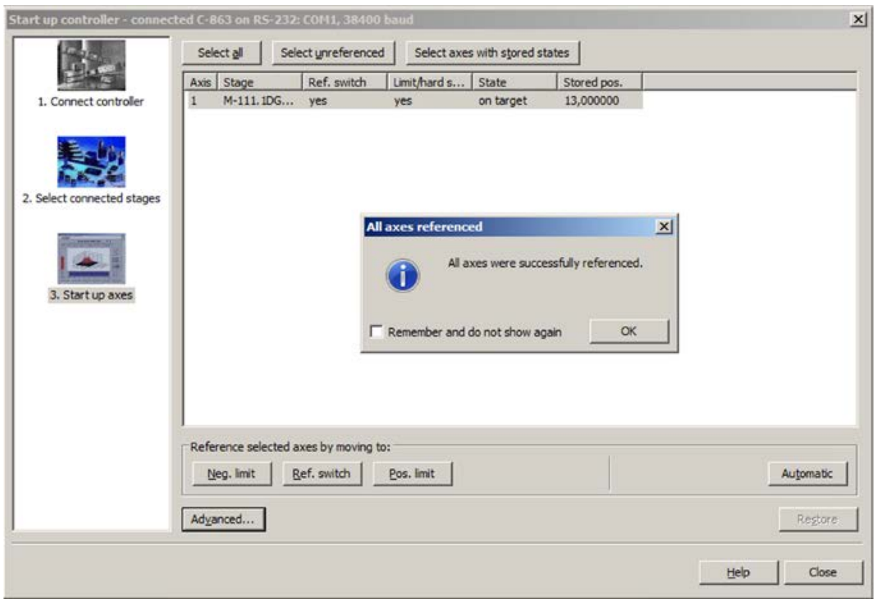
| Axis | Stage | Ref. switch | Limit/hard s... | State | Stored pos. |
|---|---|---|---|---|---|
| 1 | M-111.1DG... | yes | yes | servo off | 13,000000 |

Servo for axis 1 is off.
Do you want to switch on servo in order to reference?

Caution:
Do not switch on servo for stages without position sensor.

Figure 10: Start up controller – Start up axes

10. After a successful reference move, click on ***OK*** > ***Close***.


Start up controller - connected C-863 on RS-232: COM1, 38400 baud
1. Connect controller
2. Select connected stages
3. Start up axes
Select all
Select unreferenced
Select axes with stored states
All axes referenced
OK
Reference selected axes by moving to:
Neg. limit
Ref. switch
Pos. limit
Automatic
Advanced...
Restore
Help
Close

| Axis | Stage | Ref. switch | Limit/hard s... | State | Stored pos. |
|---|---|---|---|---|---|
| 1 | M-111.1DG... | yes | yes | on target | 13,000000 |

All axes were successfully referenced.

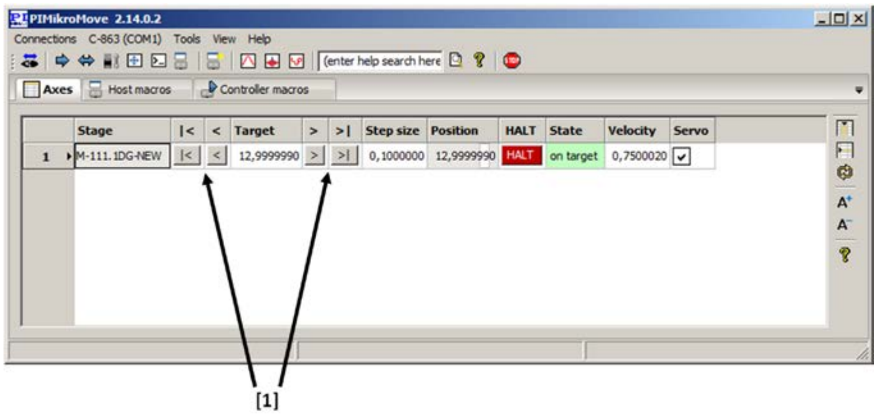
☐ Remember and do not show again

Figure 11: Start up controller – All axes referenced

The main window of PIMikroMove opens.

11. Start a few test motions of the axis.

    In the main window of PIMikroMove you can execute, for example, steps with a certain step size by clicking the corresponding arrow keys for the axis.



| | Stage | \|< | < | Target | > | >\| | Step size | Position | HALT | State | Velocity | Servo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | M-111.1DG-NEW | \|< | < | 12,9999990 | > | >\| | 0,1000000 | 12,9999990 | HALT | on target | 0,7500020 | ✓ |

*Figure 12: Main window of PIMikroMove; [1] arrow keys for motion*

# 6 Installation

## In this Chapter

## 6.1 Installing the PC Software

Communication between the C-863 and a PC is required to configure the C-863 and to command motions using the commands fo the GCS. Various PC software applications are available for this purpose.

### 6.1.1 Performing the Initial Installation

#### Accessories

- PC with a Windows operating system (XP, Vista, 7) or Linux operating system
- Product CD (included in the scope of delivery)

#### Installing the PC software on Windows

1. Start the installation wizard by double-clicking the icon or the ***setup.exe*** file in the installation directory (root directory of the CD).
2. Follow the instructions on the screen.
   You can choose between the default installation (typical) or the user-defined installation (custom). With the default installation (recommended), the following components are installed among others:
   - LabVIEW drivers
   - Dynamic program library for GCS
   - PIMikroMove

3. If you want to connect the controller to the PC via the USB interface:
   a) Start the installation of the USB drivers by clicking on ***Yes*** in the particular dialog window.
   b) Follow the instructions on the screen.

### Installing the PC software on Linux

1. Unpack the tar archive from the /linux directory of the product CD to a directory on your PC.
2. Open a terminal and go to the directory to which you have unpacked the tar archive.
3. Log on as a superuser (root rights).
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.

## 6.1.2 Installing Updates

PI is constantly improving the PC software.

➢ Always install the latest version of the PC software and the PIStages2.dat stage database.

### Prerequisite

✓ Active connection of the PC to the Internet.

– If your PC does not have an Internet connection:
  You have the A000T0032 Technical Note for the PI Update Finder on hand. You can find the document either on the product CD or in our Update Portal (http://www.update.pi-portal.ws) in the zip file for the PI Update Finder.

✓ If your PC uses a Windows operating system:

– If the PI Update Finder program is not on your product CD:
  You have downloaded the PI Update Finder from our Update Portal (http://www.update.pi-portal.ws) and unpacked it from the zip file into a directory on your PC.

– You have the A000T0028 Technical Note for the PI Update Finder on hand. You can find the document either on the product CD or on in the zip file that you have downloaded for the PI Update Finder.

✓ If your PC uses a Linux operating system:

– You have the user name and password for the C-863 at hand. Both of these can be found in the file "C-863 Releasenews_V_x_x_x.pdf" (x_x_x: version number of the CD) in the \Manuals folder on the product CD.

## Updating the PC software and PIStages2.dat on Windows

➢ Use the PI Update Finder:

– If the PC to be updated has an active connection to the Internet: Follow the instructions in the A000T0028 Technical Note (TECHNICAL_NOTE_PI_UPDATE_FINDER_xx.pdf).

– If the PC to be updated has no active connection to the Internet: Follow the instructions in the A000T0032 Technical Note.

## Updating the PC software on Linux

1. Open the PI website (http://www.pi-portal.ws).
2. Click ***Downloads***.
3. In the ***User login*** area on the left margin, enter the user name and password from the "C-663.11_Releasenews_V_x_x_x.pdf" file on the product CD.
4. Click ***Login***.
5. Click the category ***C Motion Controllers***.
6. Click ***C¬863 > C¬863.11 > software*** (if you click ***Documents***, the latest versions of the corresponding manuals are displayed).
7. Underneath the latest CD mirror, click the ***Download*** button (also contains the manuals).
8. Save the downloaded archive file on the PC.
9. Unpack the file to a separate installation directory.
10. In the directory with the unpacked files, go to the linux subdirectory.
11. Unpack the archive file in the linux directory by entering the command tar -xvpf <name of the archive file> on the console.
12. Read the accompanying information (readme file) on the software update.
13. Log onto the PC as a superuser (root rights).
14. Only install the update if it is a good idea for your application.

### Updating PIStages2.dat on Linux

1. Open the PI website (http://www.pi-portal.ws).
2. Click ***Downloads***.
3. In ***User login*** area on the left margin, enter your username and password from the "C-663.11_Releasenews_V_x_x_x.pdf" file on the product CD.
4. Click ***Login***.
5. Click on the ***General Software*** category.
6. Click on ***PI Stages***.
7. Click on ***pistages2*** or on the ***Download*** button below it.
8. Log onto the PC as a superuser (root rights).
9. Install the downloaded pistages2.dat file on the PC. You can select between the following options:
   - Save the pistages2.dat file in the /usr/local/PI/pi_gcs_translator/ directory.
   - Save the pistages2.dat file in the directory where you unpacked the Linux software from the product CD. The path is /<UnpackingDirectory>/pi_stages2_dat. In this subdirectory start the INSTALL.pi_stages2_dat script.

## 6.1.3 Installing a Custom Stage Database

With a custom stage, you will receive, if necessary, a file from PI with a custom stage database. You have to install this file on your PC so that you can load the parameter values for the custom stage in the C-863.

### Installing a custom stage database on Windows

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

   If you are working with PIMikroMove:

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

   The path where the \PI directory is located was defined during the installation of the PC software, normally C:\Documents and Settings\All Users\Application Data (Windows XP) or C:\ProgramData (Windows Vista, Windows 7).

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

***INFORMATION***

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

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

### Installing a custom stage database on Linux

1. Log onto the PC as a superuser (root rights).
2. Copy the stage database file to the /usr/local/PI/pi_gcs_translator/ directory.

## 6.2 Mounting the C-863

The C-863 can be used as bench-top device or mounted in any orientation on a surface.

***INFORMATION***

The C-863 is stackable and can be installed in a control cabinet.

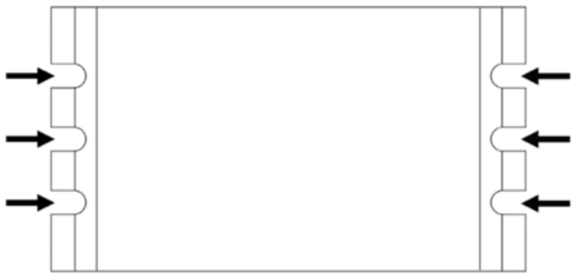


*Figure 13: C-863.11: Mounting strips with recesses (see arrows)*

### Tools and accessories

- Appropriate screws; see dimensional drawing (p. 296).
- Suitable screwdriver

### Mounting the C-863 on a surface

1. Make the necessary holes in the surface.

   The arrangement of the recesses in the mounting strips of the C-863 can be found in the dimensional drawing (p. 296).
2. Mount the C-863 to the recesses in the mounting strips with two suitable screws each per side (see figure).

## 6.3 Grounding the C-863

The C-863 is not grounded via the power supply connector.

If a potential equalization is required:

➢ Connect the screw designated with the ground connection symbol (see figure) on the rear panel of the case of the C-863 to the grounding system.

## 6.4 Connecting the Power Supply to the C-863

***NOTICE***

**Motor damage due to excessively high operating voltage!**

The output voltage on the **Motor +** and **Motor -** pins of the **DC Motor only** socket can be as high as the supply voltage provided by the power supply. Stages without PWM amplifier are connected to these pins and can be damaged by output voltage that is too high.

➢ Connect a power supply whose output voltage does **not** exceed the permissible operating voltage of the stage.

### Prerequisites

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

### Tools and accessories

- Included 15 V wide-range-input power supply (for line voltages between 100 and 240 volts alternating current voltage at 50 or 60 Hz) or other suitable power supply that supplies 15 to 30 volts direct current voltage.

### Connecting the power supply to the C-863

➢ Connect the power supply to the **15-30 VDC** connection of the C-863.

## 6.5 Connecting the Stage

*NOTICE*

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

Connecting a stage with stepper motor to a DC motor controller can cause irreparable damage.

➢ Only connect to the C-863 a stage with DC motor or voice coil drive.

*INFORMATION*

The C-863 supports stages with a PWM amplifier as well as stages without a PWM amplifier. Separate lines on the **DC Motor only** socket (p. 297) are available for both stage versions. The selection of the proper lines must be ensured via the connector of the stage. With stages from PI the proper connection is ensured.

### Prerequisite

✓ The C-863 is switched off, i. e. the power supply is **not** connected to the power socket with the power cord.

### Tools and accessories

- If the distance between the C-863 and the stage is too great:
  motor cable C-815.38, 3 m, available as an optional accessory (p. 16)

### Connecting the stage

➢ Connect the stage to the **DC Motor only** socket of the C-863.

## 6.6 Connecting the PC

The communication between the C-863 and a PC is necessary to configure the C-863 and send motion commands using the commands of the GCS. The C-863 has the following interfaces for this purpose:

- RS-232 interface
- USB interface

In this section, you will learn how to make the proper cable connections between the C-863 and a PC as well as in a daisy chain. All other steps required for establishing communication between the C-863 and PC are described in the following sections:

- "Establishing Communication via RS-232" (p. 72)
- "Establishing Communication via USB" (p. 74)
- "Establishing Communication for Networked Controllers" (p. 75)

***INFORMATION***

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

### 6.6.1 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.

#### Prerequisites

✓ 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-863 to the PC

➢ Connect the **RS-232 In** socket on the front panel of the C-863 and the RS-232 interface of the PC (a Sub-D 9(m) panel plug) to the null-modem cable.

### 6.6.2 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.

#### Prerequisites

✓ The PC has a free USB interface.

#### Tools and accessories

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

#### Connecting the C-863 to the PC

➢ Connect the USB socket of the C-863 and the USB interface of the PC with the USB cable.

### 6.6.3 Setting Up a Daisy Chain Network

***INFORMATION***

Interlinking in a daisy chain occurs in series. See also "Definition" (p. 2). Here the first controller is connected directly to the PC.

***INFORMATION***

The DIP switches of the C-863 must be set accordingly:

➢ Set a unique address for each controller in a daisy chain network. In doing so, one of the controllers must have the address 1. This controller does not have to be the one directly connected to the PC. Details see "Controller Address" (p. 69).

➢ Set the same baud rate for every controller in a daisy chain network. Details see "Baud Rate" (p. 70).

### 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. 16)

### Interlinking the controllers

- ➢ Set up the controller chain. For this purpose, always connect the **RS-232 Out** connection of the previous controller via the network cable to the **RS-232 In** connection of the subsequent controller.
- ➢ Connect the first controller of the chain to the PC.
  - Use the RS-232 interface (p. 60).

  **or**

  - Use the USB interface (p. 61).

***INFORMATION***

A C-863.11 can be operated in a common daisy chain network with the following controllers:

- C-663.11 Mercury Step stepper motor controller
- PILine® piezomotor controllers of the C-867 series
- E-861 NEXACT® controller

## 6.7 Connecting an Analog Joystick

***INFORMATION***

You can connect one axis of an analog joystick to the **Joystick** socket. With the joystick axis you can control the velocity of the stage connected to the C-863.

You have the following connection options:

- Pin 4: 0 to 3.3 V input voltage (commandable as axis 1 of joystick device 1)

or

- Pin 2: -10 to 10 V input voltage (commandable as axis 1 of joystick device 2)

A joystick button can be connected to pin 6 of the **Joystick** socket (commandable as button 1 of joystick device 1).

***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 source of the joystick.

You can use a C-819.20Y Y cable to connect two C-863 to a C-819.20 joystick. In this case, the joystick is powered by the C-863 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. 16):
  - 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
- Alternatively: joystick that supplies -10 to 10 V

### Connecting an analog joystick

➢ Connect the joystick to the **Joystick** socket of the C-863:

  – If you want to operate a C-819.20 joystick only with this controller, 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 joystick is powered 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 enabled 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.
- If you want to use a joystick that supplies -10 to 10 V, connect it to pin 2 of the **Joystick** socket.

## 6.8 Connecting Digital In- and Outputs

The digital inputs and outputs on the **I/O** socket of the C-863 can be used as follows:

- Outputs: Triggering of external devices; see "Digital Output Signals" (p. 97).
- Inputs: Use in macros (p. 109) and/or as source for the reference and limit switch signals of the axis (p. 109)

### 6.8.1 Connecting the 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. 16)
- Device to be triggered having digital input for TTL signals

#### Connecting a device to be triggered

➢ Connect an appropriate device to one of the pins 5, 6, 7 or 8 of the **I/O** socket of the C-863.

## 6.8.2 Connecting the Inputs

***INFORMATION***

Digital input signals can be fed via pins 1, 2, 3 and 4 of the **I/O** socket into the C-863.

***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. 16).
  - 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. 16).

### 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-863.

## 6.9 Connecting Analog Signal Sources

The analog inputs on the **I/O** socket of the C-863 can be used as follows:

- Use in macros (p. 112): Details and examples of macros are found in "Controller Macros" (p. 121).
- Scanning applications with PIMikroMove (see PIMikroMove manual)

***INFORMATION***

Analog input signals can be fed into the C-863 via pins 1, 2, 3 and 4 of the **I/O** socket.

***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. 16).

### Connecting an analog signal source

➢ Connect an appropriate signal source to one of the pins 1, 2, 3 or 4 of the **I/O** socket of the C-863.

# 7 Start-Up

## In this Chapter

## 7.1 General Notes on Start-Up

*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 mechanical system.

➢ Prevent motions in open-loop operation.

➢ If motions in open-loop operation are necessary:

  – Set the control value with the SMO command so that the axis moves with low velocity.
  – Stop the axis in time. For this purpose, use the #24, STP or HLT command, or set the control value to zero with the SMO command.

➢ Do not disable the evaluation of the limit switches by the C-863 via parameter setting.

➢ Check the function of the limit switches at about 10 % to 20 % of the maximum velocity.

➢ In the event of a malfunction of the limit switches, stop the motion immediately.

## 7.2 Adapting the DIP Switch Settings

### 7.2.1 General Procedure

***INFORMATION***

Changed DIP switch settings become effective after the C-863 is switched on.

➢ If you have changed the DIP switch settings while the C-863 was switched on, switch the C-863 off and back on again to activate the new settings.

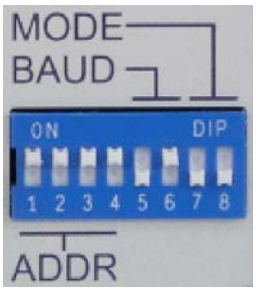


*Figure 14: DIP switches: switch up = ON; switch down = OFF*

| Switches | Function |
|---|---|
| 1 to 4 | Controller address (p. 69); 16 possible combinations |
| 5 and 6 | Baud rate (p. 70) |
| 7 | Logic level of the limit switches (p. 70) |
| 8 | Update mode (p. 71) |

#### Prerequisite

✓ The C-863 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 given in the following tables.

### 7.2.2 Controller Address

| Address* | S1 | S2 | S3 | S4 |
|---|---|---|---|---|
| **1** | **ON** | **ON** | **ON** | **ON** |
| 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 settings are shown in bold.

***INFORMATION***

A unique address must be set for each controller in a daisy chain network. In doing so, 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 the address 1, if

- it is to be used in PIMikroMove.
- it is to be used in LabVIEW.
- it is to be addressed with the PITerminal without specifying the target address; in this case, in the responses of the C-863 target and sender addresses (p. 146) are omitted as well.

### 7.2.3 Baud Rate

| **Baud rate*** | **S5** | **S6** |
|---|---|---|
| 9600 | ON | ON |
| 19200 | ON | OFF |
| **38400** | **OFF** | **ON** |
| 115200 | OFF | OFF |

*Factory settings are shown in bold.

***INFORMATION***

The same baud rate must be set for every controller in a daisy chain network.

### 7.2.4 Logic Level of the Limit Switches

| **Limit switches (hardware setting)*** | **S7** |
|---|---|
| Active low | ON |
| **Active high** | **OFF** |

*Factory settings are shown in bold.

***INFORMATION***

The C-863 can be adapted via DIP switch 7 and the ***Limit Mode*** parameter (ID 0x18) to the logic level of the limit switches of the stage connected (active high or active low). In order for the stage to be able to move, the DIP switch **and** parameter settings must correspond to the logic level of the limit switches.

DIP switch 7 is preset for using active high signals (e. g. for DC motor stages from PI). If you select your PI stage from a stage database in the PC software from PI, the ***Limit Mode*** parameter is already set correctly.

### 7.2.5 Update Mode

| **Update mode** | **S8** |
|---|---|
| Firmware update | ON |
| **Normal operation** | **OFF** |

*Factory settings are shown in bold.

***INFORMATION***

If the C-863 is in the firmware update mode (DIP switch 8 in "ON" (upper) position), all LEDs remain deactivated after switching on the C-863.

## 7.3 Switching on the C-863

***INFORMATION***

The C-863 is intended for closed-loop operation with incremental position sensors (servo mode On). After switching-on, open-loop operation is enabled by default (servo mode Off).

- ➢ Get the current operating mode with the `SVO?`, `#4` or `SRG?` commands.
- ➢ Enable closed-loop operation with the `SVO` command.
- ➢ If necessary, program a start-up macro that starts the C-863 via the `SVO` command in closed-loop operation; see "Setting up a start-up macro" (p. 131).
- ➢ Prevent motions in open-loop operation.

### Prerequisites

- ✓ You have read and understood the General Notes on Start-Up (p. 67).
- ✓ The C-863 has been installed properly (p. 53).
- ✓ You have set the DIP switches of the C-863 in accordance with your application (p. 68).

### Switching on the C-863

➢ Connect the power cord of the power supply with the power socket.

The C-863 copies information from the nonvolatile memory to the volatile memory.

The **STA** LED on the front panel of the C-863 displays the state of the C-863:

– green: C-863 is ready for normal operation

– off: If DIP switch 8 is in the "ON" (upper) position, the C-863 is in firmware update mode. Otherwise the C-863 might be defective.

➢ If DIP switch 8 is in the "OFF" (lower) position and the **STA** LED does not light after switching-on, contact our customer service department (p. 291).

## 7.4 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. Details see "Controller Address" (p. 69).

### 7.4.1 Establishing Communication via RS-232

#### Prerequisites

✓ You have read and understood the General Notes on Start-Up (p. 67).

✓ The C-863 is connected to the RS-232 interface of the PC (p. 60).

✓ You have made the following settings with the respective DIP switches prior to switching on the C-863 (p. 68):

– controller address = 1

– appropriate baud rate

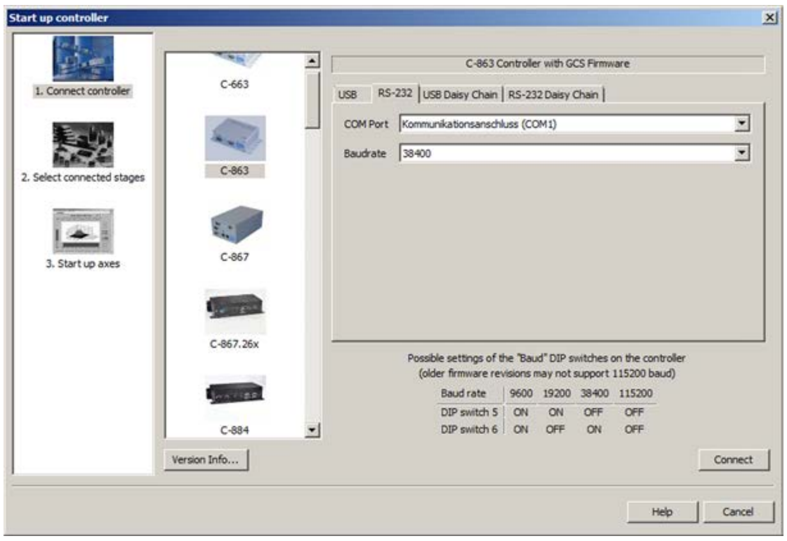
- ✓ The C-863 is switched on (p. 71).
- ✓ The PC is switched on.
- ✓ The required software is installed on the PC (p. 53).
- ✓ You have read and understood the manual of the used PC software. The software manuals are found on the product CD.

## Establishing communication

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.



C-863 Controller with GCS Firmware

COM Port: Kommunikationsanschluss (COM1)

Baudrate: 38400

Possible settings of the "Baud" DIP switches on the controller
(older firmware revisions may not support 115200 baud)

| Baud rate | 9600 | 19200 | 38400 | 115200 |
|---|---|---|---|---|
| DIP switch 5 | ON | ON | OFF | OFF |
| DIP switch 6 | ON | OFF | ON | OFF |

*Figure 15: Start up controller – Connect controller*

2. Select ***C-863*** in the field for controller selection.
3. Select the ***RS-232*** tab on the right side of the window.
4. In the ***COM Port*** field, select the COM port of the PC to which you have connected the C-863.
5. In the ***Baudrate*** field, set the value that is set with DIP switches 5 and 6 of the C-863.

   This adapts the baud rate of the PC to the baud rate of the C-863.
6. Click ***Connect*** to establish communication.

If communication has been successfully established, the ***Start up controller*** window switches to the ***Select connected stages*** step.

## 7.4.2 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.

### Prerequisites

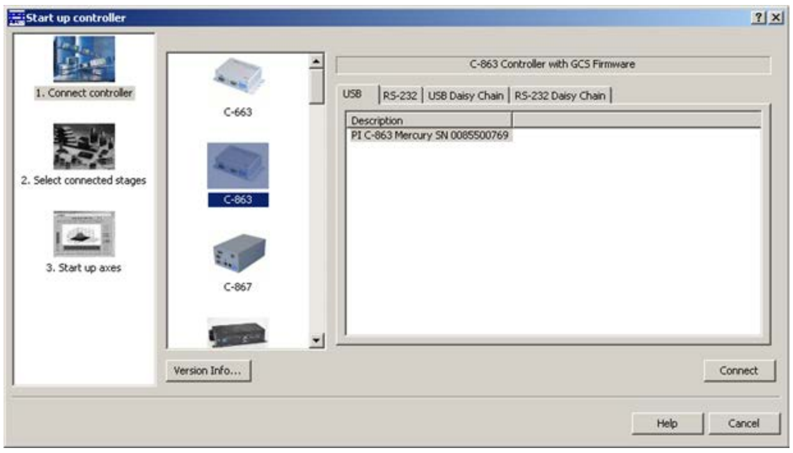
- ✓ You have read and understood the General Notes on Start-Up (p. 67).
- ✓ The C-863 is connected to the USB interface of the PC (p. 61).
- ✓ Prior to switching on the C-863, you have set the DIP switches for the controller address to the address 1 (p. 68).
- ✓ The C-863 is switched on (p. 71).
- ✓ The PC is switched on.
- ✓ The required software and USB drivers are installed on the PC.
- ✓ You have read and understood the manual of the used PC software. The software manuals are found on the product CD.

### Establishing communication

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.



PI C-863 Mercury SN 0085500769

Figure 16: Start up controller – Connect controller

2. Select ***C-863*** in the field for controller selection.
3. Select the ***USB*** tab on the right side of the window.
4. On the ***USB*** tab, select the connected C-863.
5. Click ***Connect*** to establish communication.

If communication has been successfully established, the ***Select connected stages*** dialog opens.

➢ If communication could not be established, look for a solution to the problem in "Troubleshooting" (p. 285).

### 7.4.3 Establishing Communication for a Networked Controller

The procedure for PIMikroMove and for PITerminal is described in the following.

***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. Details see "Target and Sender Address" (p. 146).

➢ 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.

## Prerequisites

✓ You have read and understood the General Notes on Start-Up (p. 67).

✓ You have set up a daisy chain network (p. 61).

✓ Prior to switching-on you have properly set on each networked C-863 the DIP switches for the controller address (p. 69) and the baud rate (p. 70).

✓ Every controller in the daisy chain network is switched on (p. 71).

✓ The PC is switched on.

✓ The required software is installed on the PC (p. 53).

✓ 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. 53).

✓ You have read and understood the manual of the used PC software. The software manuals are found on the product CD.

## Establishing communication with PIMikroMove

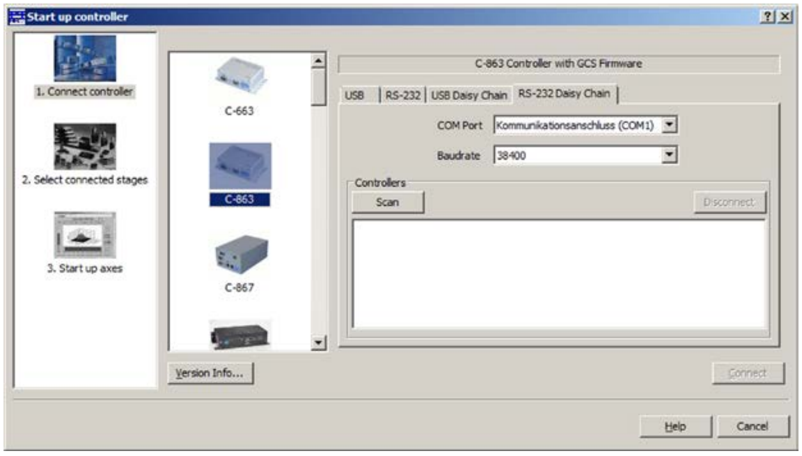
1. Start PIMikroMove.

   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-863.11 with the controller address 1 and a C-663.11 with the controller address 2. If you want to connect the C-863.11 first, select ***C-863***.

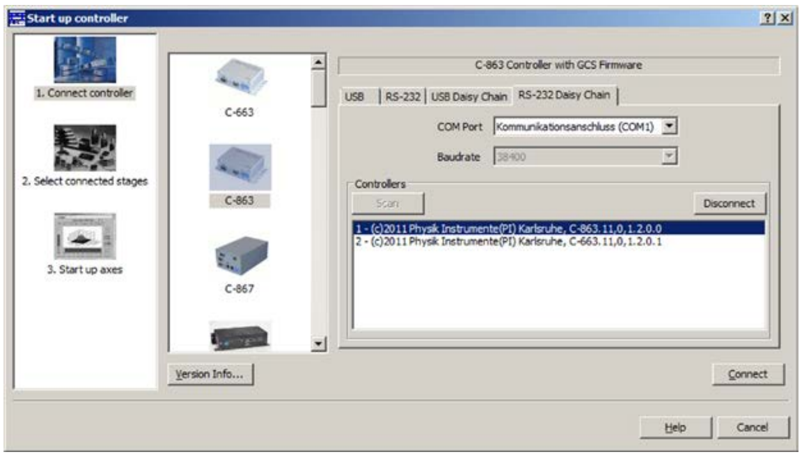


COM Port: Kommunikationsanschluss (COM1)

Baudrate: 38400

3. Select the appropriate tab on the right 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:
     - In the ***COM Port*** field, select the PC COM port to which you have connected the C-863.
     - In the ***Baudrate*** field, set the value which is set with DIP switches 5 and 6 of the C-863.
   – ***USB Daisy Chain*** tab:
     - In the top section of the tab, select the C-863 connected.

5. In the bottom section of the tab, click on the ***Scan*** button to list every controller in the daisy chain network.



C-863 Controller with GCS Firmware

COM Port: Kommunikationsanschluss (COM1)

Baudrate: 38400

Controllers

1 - (c)2011 Physik Instrumente(PI) Karlsruhe, C-863.11,0,1.2.0.0
2 - (c)2011 Physik Instrumente(PI) Karlsruhe, C-663.11,0,1.2.0.1

6. Select a controller from the list. The selection must match the controller type that you selected in step 2.

7. Click ***Connect*** to establish communication with the controller selected.

   If communication has been successfully established, the ***Select connected stages*** dialog opens.

   – Proceed further as described in "Starting Motions" (p. 81).

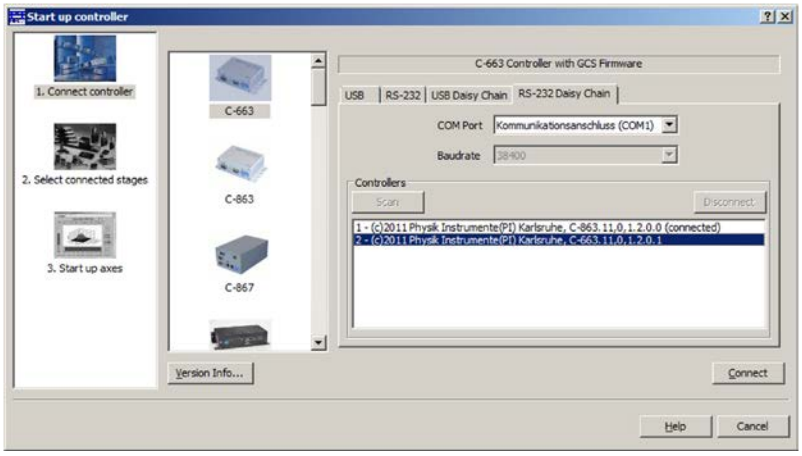
   or

   – Close the ***Select connected stages*** window by clicking on the ***Cancel*** button.

8. If you want to connect an additional controller of the daisy chain network, select the ***Connections > New...*** menu item in the main window.

9. Execute again steps 2, 6 and 7 in the order indicated.

In the following figure, the ***C-663*** is to be connected as well.



C-663 Controller with GCS Firmware

COM Port: Kommunikationsanschluss (COM1)

Baudrate: 38400

Controllers

1 - (c)2011 Physik Instrumente(PI) Karlsruhe, C-863.11,0,1.2.0.0 (connected)
2 - (c)2011 Physik Instrumente(PI) Karlsruhe, C-663.11,0,1.2.0.1

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:

➢ In the main window, select the ***Connections > Close*** menu item for the corresponding controller.

## Establishing communication with PITerminal

***INFORMATION***

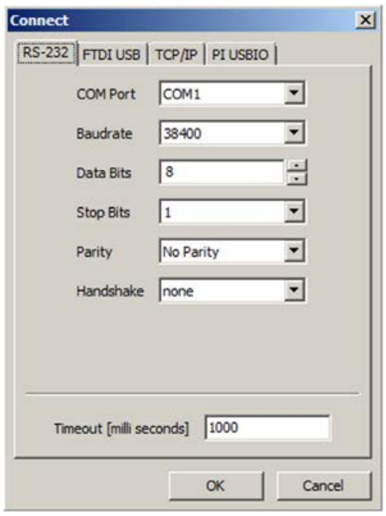
Via the ***Mercury*** button, PITerminal supports controllers with older firmware versions that are not compatible with GCS.

➢ Make sure that the **Mercury** button is ***not*** enabled in PITerminal.

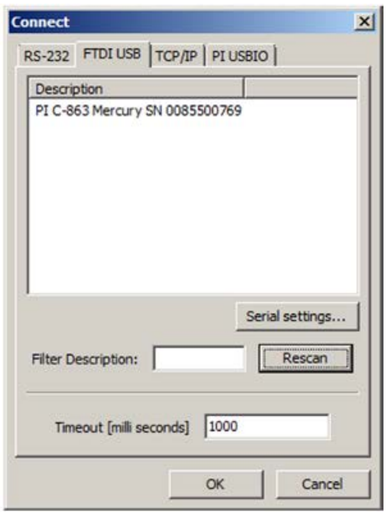
1. Start PITerminal.
2. Click on ***Connect…***.

   The ***Connect*** window opens.

3. In the ***Connect*** window, select the ***RS-232*** or ***FTDI USB*** tab, depending on via which interface you have connected the first controller in the chain to the PC.



COM Port COM1
Baudrate 38400
Data Bits 8
Stop Bits 1
Parity No Parity
Handshake none
Timeout [milli seconds] 1000

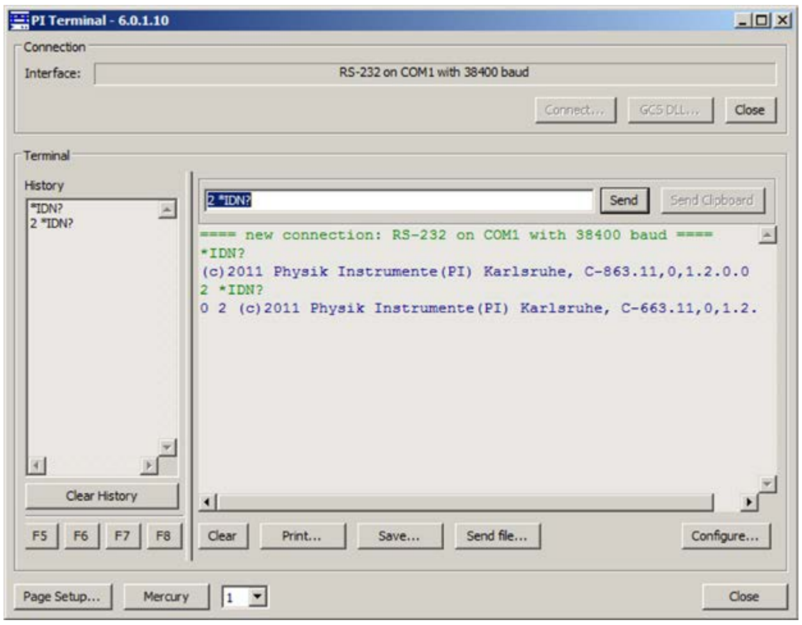


Description
PI C-863 Mercury SN 0085500769
Filter Description:
Timeout [milli seconds] 1000

4. Make the settings for the interface in the selected tab:
   - ***RS-232*** tab:
     - In the ***COM Port*** field, select the PC COM port to which you have connected the C-863.
     - In the ***Baudrate*** field, set the value which is set with DIP switches 5 and 6 of the C-863.
   - ***FTDI USB*** tab:
     - Select the connected C-863.
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 comprises one C-863.11 with the controller address 1 and one C-663.11 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.

Further information see "Target and Sender Address" (p. 146).


PI Terminal - 6.0.1.10
Connection
Interface: RS-232 on COM1 with 38400 baud
Connect... GCS DLL... Close
Terminal
History
*IDN?
2 *IDN?
2 *IDN?
Send
Send Clipboard
Clear History
F5 F6 F7 F8
Clear Print... Save... Send file... Configure...
Page Setup... Mercury 1 Close

```
==== new connection: RS-232 on COM1 with 38400 baud ====
*IDN?
(c)2011 Physik Instrumente(PI) Karlsruhe, C-863.11,0,1.2.0.0
2 *IDN?
0 2 (c)2011 Physik Instrumente(PI) Karlsruhe, C-663.11,0,1.2.
```

## 7.5 Starting Motions

In the following, PIMikroMove is used to move the stage. Here the program guides you through the following steps so that you do not have to deal with the respective GCS commands:

- Adapting the parameter settings of the C-863 to the connected stage by loading a parameter set from a stage database
- Switching on the servo mode (closed-loop operation)
- Executing a reference move; details see "Reference Point Definition" (p. 38).

***NOTICE***

**Selection of an incorrect stage type**

Selecting an incorrect stage type in the PC software can cause damage to the stage.

➢ Make sure that the stage type selected in the PC software matches the stage connected.

*NOTICE*

**Oscillations!**

Unsuitable settings made to the servo-control parameters of the C-863 can cause the stage to oscillate. Oscillations can damage the stage and/or the load affixed to it.

- Secure the stage and all loads adequately.
- If the stage is oscillating (unusual operating noise), immediately switch off the servo mode or disconnect the C-863 from the power source.
- Only switch on the servo mode after you have modified the servo-control parameter settings of the C-863; see "Optimizing Servo-Control Parameters" (p. 90).
- If, due to a very high load, oscillations occur already during the reference move, follow the instructions for the reference move in "Troubleshooting" (p. 285).

***INFORMATION***

Parameter values from the stage database are only loaded to the volatile memory of the controller.

***INFORMATION***

The C-863 has a nonvolatile memory for parameter values. Therefore, after switching-on, the correct parameter settings for the stage connected may already be loaded.

- However, if you have loaded a parameter set from the stage database and overwritten the original settings of the C-863 in the volatile memory in the process, avoid saving the new settings in the nonvolatile memory of the C-863. The original settings are active again after the C-863 has been switched off and on again or been rebooted.

***INFORMATION***

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

1. Check during the ***Start up axes*** step whether the correct stage type is at mid-range of the window in the ***Stage*** column.
2. If the stage type is not correct, click ***Select connected stages*** in the left area of the ***Start up controller*** window for changing the stage type.

## Prerequisites

- ✓ You have read and understood the General Notes on Start-Up (p. 67).
- ✓ PIMikroMove is installed on the PC (p. 53).
- ✓ You have read and understood the PIMikroMove manual. The manual is found on the product CD.
- ✓ You have installed the latest version of the *PIUserStages2.dat* stage database on the PC (p. 55).
- ✓ If PI has provided you with a custom stage database for your stage, then you have installed this database on your PC (p. 56).
- ✓ You have installed the stage in the same manner as it will be used in your application (corresponding load and orientation).
- ✓ You have connected the stage to the C-863 (p. 60).
- ✓ You have established communication between the C-863 and the PC with PIMikroMove via TCP/IP, RS-232 or USB (p. 74).

## Starting motions with PIMikroMove

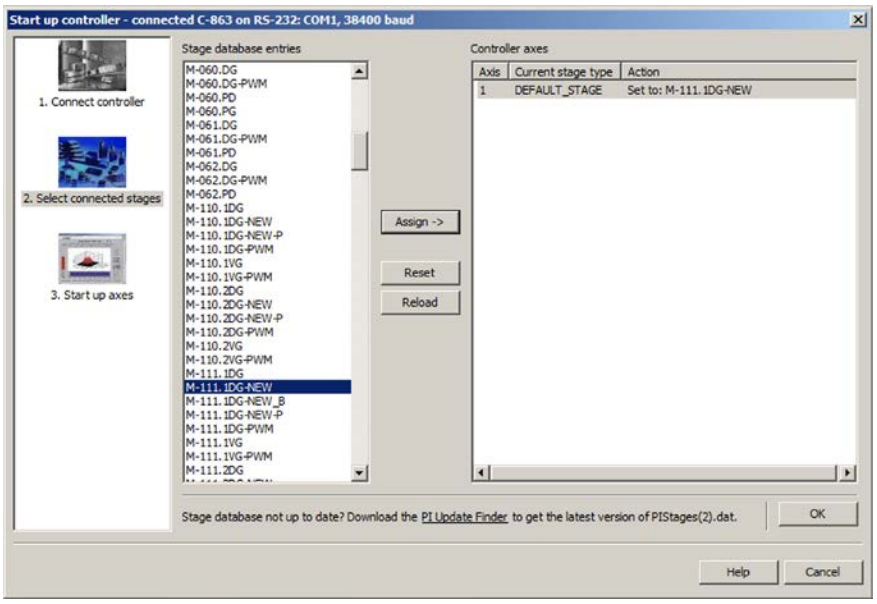
1. If in PIMikroMove the ***Select connected stages*** step is displayed, select the stage type of the stage connected:

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

   – Click on ***OK***.

   If the listed stage type is not correct:

   a) Mark the stage type in the ***Stage database entries*** list.
   b) Click ***Assign***.
   c) Confirm the selection with ***OK*** to load the parameter settings for the selected stage type from the stage database to the volatile memory of the C-863.


Start up controller - connected C-863 on RS-232: COM1, 38400 baud

Stage database entries

M-060.DG
M-060.DG-PWM
M-060.PD
M-060.PG
M-061.DG
M-061.DG-PWM
M-061.PD
M-062.DG
M-062.DG-PWM
M-062.PD
M-110.1DG
M-110.1DG-NEW
M-110.1DG-NEW-P
M-110.1DG-PWM
M-110.1VG
M-110.1VG-PWM
M-110.2DG
M-110.2DG-NEW
M-110.2DG-NEW-P
M-110.2DG-PWM
M-110.2VG
M-110.2VG-PWM
M-111.1DG
M-111.1DG-NEW
M-111.1DG-NEW_B
M-111.1DG-NEW-P
M-111.1DG-PWM
M-111.1VG
M-111.1VG-PWM
M-111.2DG

Controller axes

| Axis | Current stage type | Action |
|---|---|---|
| 1 | DEFAULT_STAGE | Set to: M-111.1DG-NEW |

Stage database not up to date? Download the PI Update Finder to get the latest version of PIStages(2).dat.

*Figure 17: Start up controller – Select connected stages*

After you have clicked ***OK***, the ***Start up controller*** window goes to the ***Start up axes*** step.

2. In the ***Start up axes*** step, execute the reference move for the axis so that the controller knows the absolute axis position:

   – If you want to start the reference move to the reference point switch, click on ***Ref. switch***.

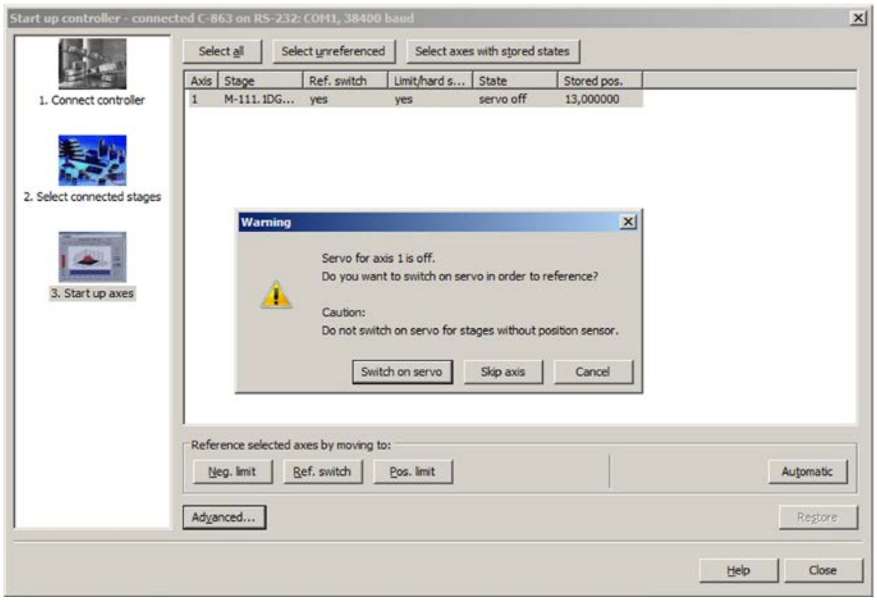
   – If you want to start the reference move to the negative limit switch, click on ***Neg. limit***.

   – If you want to start the reference move to the positive limit switch, click on ***Pos. limit***.

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

   – Switch on the servo mode by clicking on the ***Switch on servo*** button (closed-loop operation).

The axis executes the reference move.



Start up controller - connected C-863 on RS-232: COM1, 38400 baud

| Axis | Stage | Ref. switch | Limit/hard s... | State | Stored pos. |
|---|---|---|---|---|---|
| 1 | M-111.1DG... | yes | yes | servo off | 13,000000 |

Warning

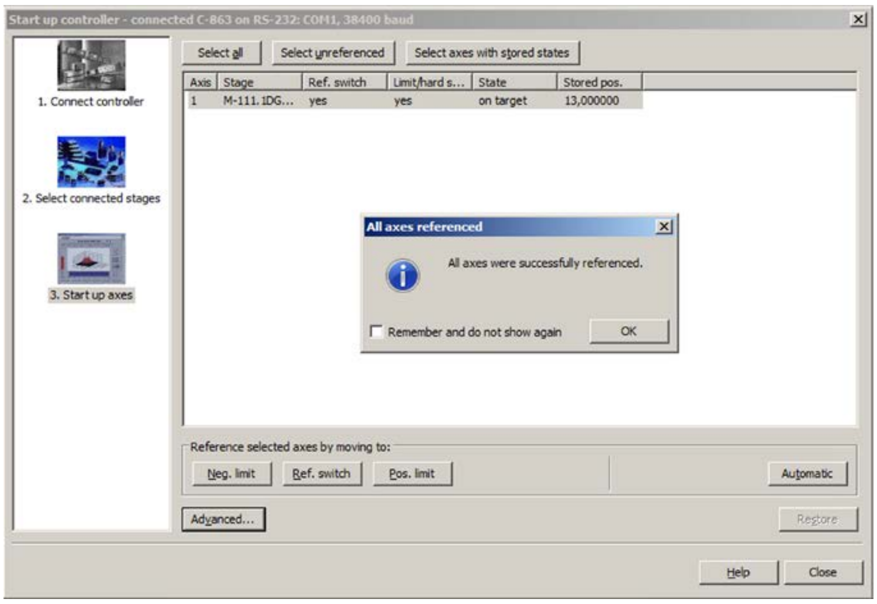
Servo for axis 1 is off.
Do you want to switch on servo in order to reference?

Caution:
Do not switch on servo for stages without position sensor.

Reference selected axes by moving to:

Figure 18: Start up controller – Start up axes

3. After a successful reference move, click ***OK > Close***.



Start up controller - connected C-863 on RS-232: COM1, 38400 baud

| Axis | Stage | Ref. switch | Limit/hard s... | State | Stored pos. |
|---|---|---|---|---|---|
| 1 | M-111.1DG... | yes | yes | on target | 13,000000 |

All axes referenced

All axes were successfully referenced.

☐ Remember and do not show again

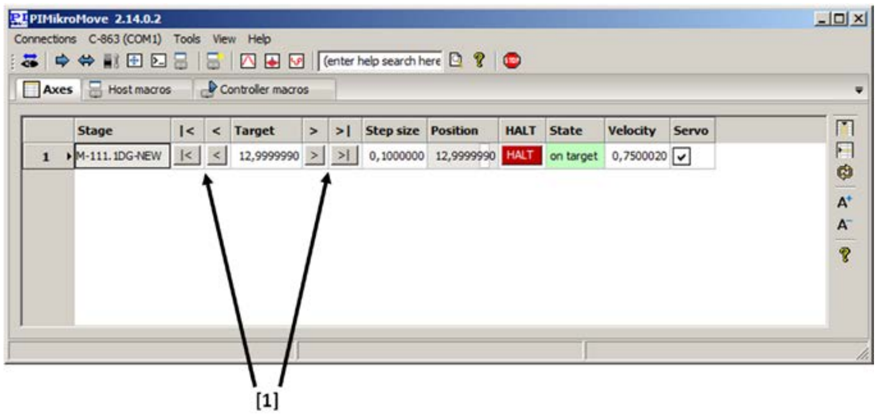
Reference selected axes by moving to:

Figure 19: Start up controller – All axes referenced

The main window of PIMikroMove opens.

4. Start a few test motions of the axis.

   In the main window of PIMikroMove you can execute, for example, steps with a certain step size by clicking the corresponding arrow keys for the axis.



| | Stage | \|< | < | Target | > | >\| | Step size | Position | HALT | State | Velocity | Servo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | M-111.1DG-NEW | \|< | < | 12,9999990 | > | >\| | 0,1000000 | 12,9999990 | HALT | on target | 0,7500020 | ✓ |

*Figure 20: Main window of PIMikroMove; [1] arrow keys for motion*

## 7.6 Recording the Step Response

With the recording of the step response, you determine the settling behavior of the stage in closed-loop operation. The procedure for PIMikroMove is described in the following.

### Prerequisite

- ✓ With PIMikroMove you have started initial motions (p. 81).
- ✓ All devices are still ready for operation.

## Measuring the step response

1. In the main window of PIMikroMove, open the ***Data Recorder*** window via the ***C-863 > Show/Hide data recorder*** menu item.

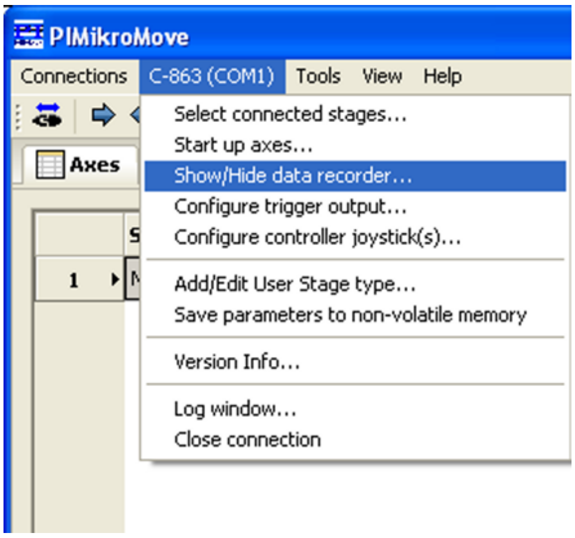


Figure 21: PIMikroMove: Show/Hide data recorder

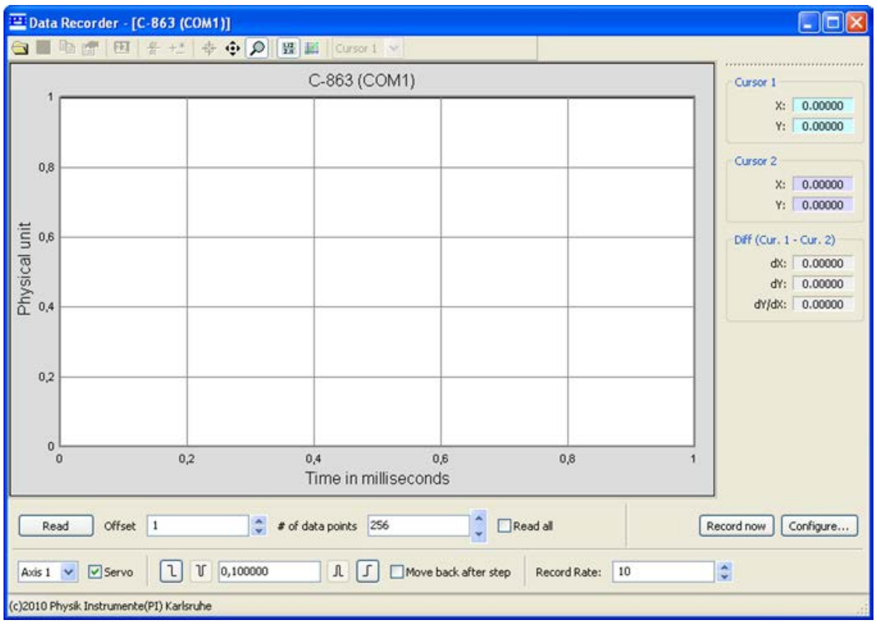


Figure 22: PIMikroMove: Data recorder

2. With the ***Servo*** check box, make sure that the servo mode is switched on.

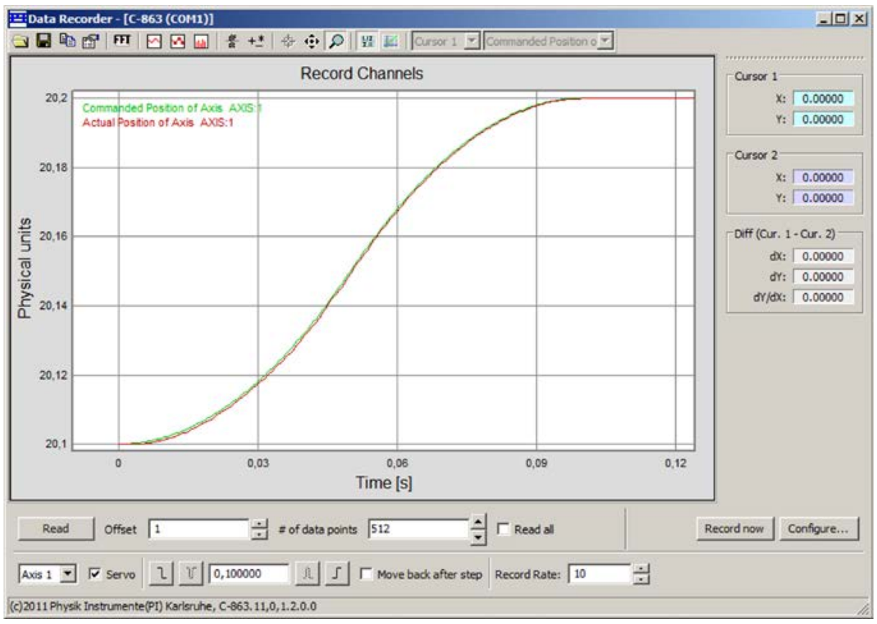
   – If the ***Servo*** check box is not checked, the servo mode is switched off. Check the check box to switch on the servo mode.

3. Configure the data recorder.
    a) In the ***Data Recorder*** window, adapt, if required, the settings like the amplitude of the step to be executed, the record table rate and for the graph, the number of data points to be read.
    b) Click on the ***Configure…*** button and make sure that "Commanded Position of Axis" and "Actual Position of Axis" are selected as quantities to be recorded.

    Details see PIMikroMove user manual (SM148E).
4. Start the step in positive direction, as well as the recording, by clicking on the button.

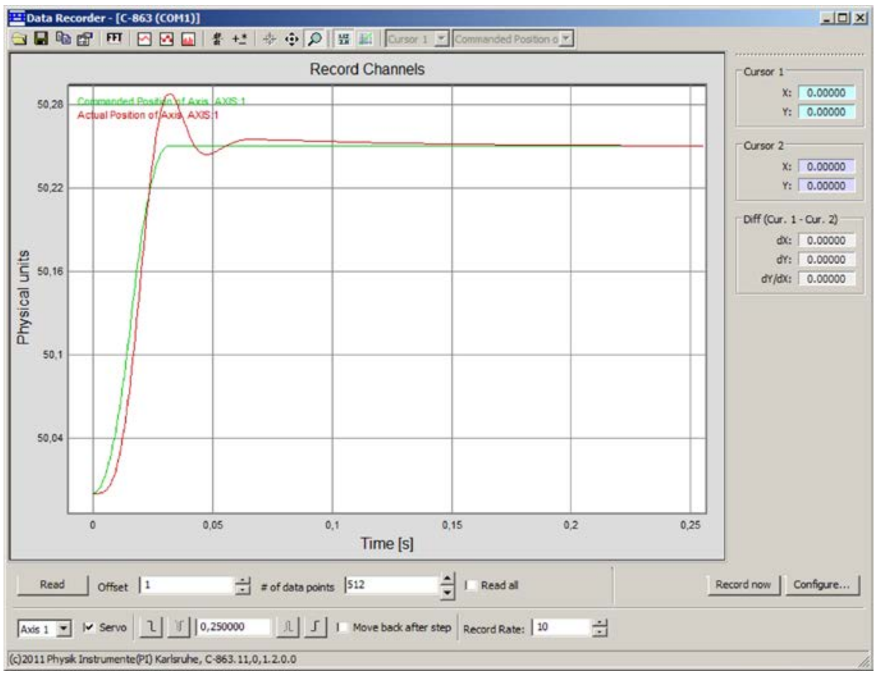
    The axis executes the step and the step response is recorded and displayed graphically.
5. Check the displayed step response on the basis of the following figures.
    a) If the result is satisfactory (i.e. minimum overshoot, settling time not too long), you already have optimal parameter settings. In this case, no further action is necessary.
    b) If the result is not satisfactory, change the settings of the P-I-D controller (p. 90).

## Examples



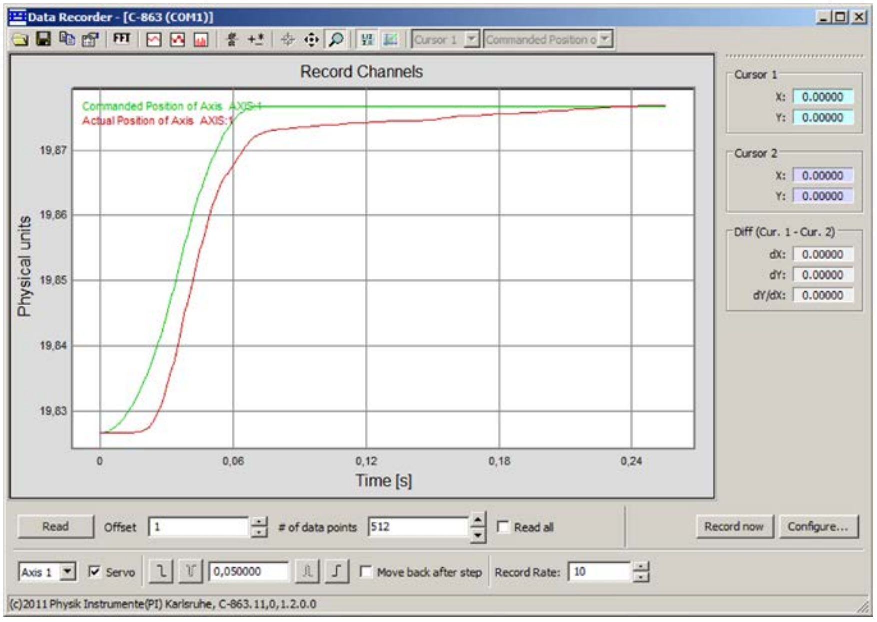
Cursor 1 X: 0.00000 Y: 0.00000
Cursor 2 X: 0.00000 Y: 0.00000
Diff (Cur. 1 - Cur. 2) dX: 0.00000 dY: 0.00000 dY/dX: 0.00000
Offset 1 # of data points 512 ☐ Read all
Axis 1 ☑ Servo 0,100000 ☐ Move back after step Record Rate: 10

Figure 23: PIMikroMove: Data recorder with graphical display of an optimal settling behavior (commanded and actual position coincident)



Cursor 1 X: 0.00000 Y: 0.00000
Cursor 2 X: 0.00000 Y: 0.00000
Diff (Cur. 1 - Cur. 2) dX: 0.00000 dY: 0.00000 dY/dX: 0.00000
Offset 1 # of data points 512 ☐ Read all
Axis 1 ☑ Servo 0,250000 ☐ Move back after step Record Rate: 10

Figure 24: PIMikroMove: Data recorder with graphical display of an overshoot



Cursor 1 X: 0.00000 Y: 0.00000

Cursor 2 X: 0.00000 Y: 0.00000

Diff (Cur. 1 - Cur. 2) dX: 0.00000 dY: 0.00000 dY/dX: 0.00000

Offset 1 # of data points 512 ☐ Read all

Axis 1 ☑ Servo 0,050000 ☐ Move back after step Record Rate: 10

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

*Figure 25: PIMikroMove: Data recorder with graphical display of a to slow step-and-settle*

## 7.7 Optimizing the Servo-Control Parameters

Adjusting the P-I-D controller optimizes the dynamic properties of the system (overshoot and settling time). The optimum P-I-D controller setting depends on your application and your requirements.

As a rule, optimization is done empirically and involves the following parameters. Details see "Servo Algorithm and Other Control Value Corrections" (p. 27):

- ***P-Term*** (0x1)
- ***I-Term*** (0x2)
- ***D-Term*** (0x3)
- ***I-Limit*** (0x4)

The behavior of the stage is monitored under various values in closed-loop operation.

In the following, PIMikroMove is used for optimizing the P-I-D servo-control parameters.

## Prerequisite

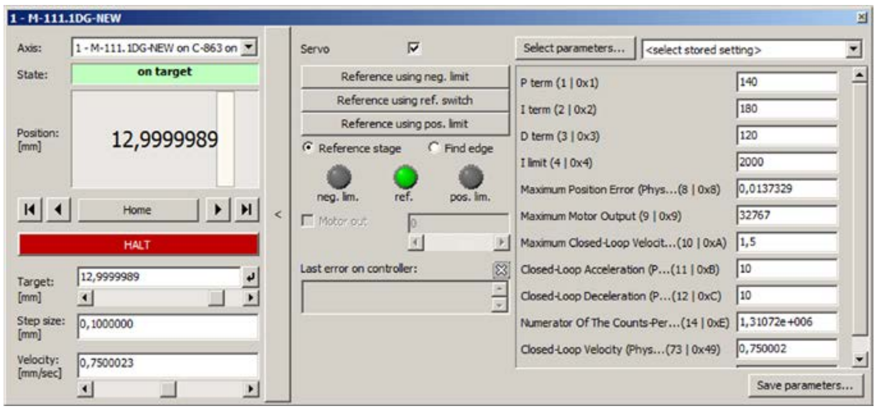
✓ The recorded step response (p. 86) of the stage has not turned out satisfactorily.

or

✓ The stage oscillates (unusual operating noise) with the current servo-control parameters.

## Adjusting the P-I-D controller

1. In the main window of PIMikroMove, open the single axis window for the connected stage by selecting the stage in the ***View > Single Axis Window*** menu.
2. Expand the view of the single axis window by clicking on the **>** button at the right edge of the window.



| | |
|---|---|
| Axis: | 1 - M-111.1DG-NEW on C-863 on |
| State: | on target |
| Position: [mm] | 12,9999989 |
| Target: [mm] | 12,9999989 |
| Step size: [mm] | 0,1000000 |
| Velocity: [mm/sec] | 0,7500023 |
| Last error on controller: | |

| | |
|---|---|
| P term (1 \| 0x1) | 140 |
| I term (2 \| 0x2) | 180 |
| D term (3 \| 0x3) | 120 |
| I limit (4 \| 0x4) | 2000 |
| Maximum Position Error (Phys...(8 \| 0x8) | 0,0137329 |
| Maximum Motor Output (9 \| 0x9) | 32767 |
| Maximum Closed-Loop Velocit...(10 \| 0xA) | 1,5 |
| Closed-Loop Acceleration (P...(11 \| 0xB) | 10 |
| Closed-Loop Deceleration (P...(12 \| 0xC) | 10 |
| Numerator Of The Counts-Per...(14 \| 0xE) | 1,31072e+006 |
| Closed-Loop Velocity (Phys...(73 \| 0x49) | 0,750002 |

*Figure 26: PIMikroMove: Expanded single axis window*

3. Enter new values for the servo-control parameters:
   a) If the parameter to be modified is not included in the list on the right side of the window, click on ***Select parameters...*** and add it to the list.
   b) Type the new parameter value into the appropriate input field in the list.
   c) Press the Enter key of the PC to transfer the parameter value to the volatile memory of the controller. When this is done, the entry changes its color from blue to black.
4. In the ***Data Recorder*** window, record the step response (p. 86) of the stage again.

If the result is not satisfactory:

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

If you are satisfied with the result and want to keep the new servo-control parameter settings:

➢ Save the new settings. You have the following options:

  – Save a new parameter set in the PI_UserStages2.dat stage database on the PC. See "Creating or Modifying a Stage Type" (p. 268).

  – Transfer the current values of **all** parameters from the volatile to the nonvolatile memory of the C-863 (p. 267).

# 8 Operation

## In this Chapter

## 8.1 Motion Errors

### 8.1.1 Protective Functions of the C-863

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

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 a motion error occurs, the C-863 reacts as follows to protect the system from damage:

- Servo mode is switched off for the axis in question.
- If present, the brake is activated for the axis in question.
- All motions are stopped.
- Error code -1024 is set.

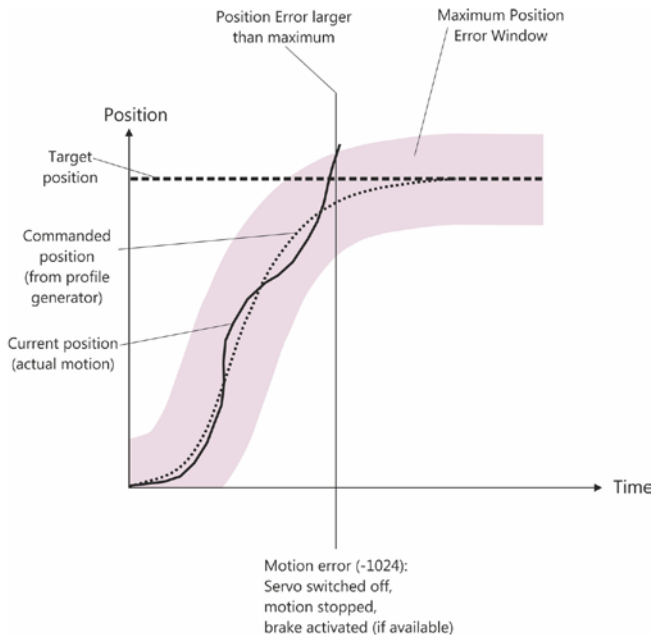

Position Error larger than maximum
Maximum Position Error Window
Position
Target position
Commanded position (from profile generator)
Current position (actual motion)
Time
Motion error (-1024):
Servo switched off,
motion stopped,
brake activated (if available)

*Figure 27: Behavior in case of motion errors*

### 8.1.2 Re-establishing Readiness for Operation

*NOTICE*

**Unintentional motions after brake deactivated!**

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

➢ Secure the stage against unintentional motions before you deactivate the brake by command!

#### Re-establishing readiness for operation

1. Send the `ERR?` command to read out the error code.

   If a motion error occurred, error code -1024 is output. `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. Switch on the servo mode for the axis in question with the `SVO` command (p. 233).

   When switching on the servo mode, the target position is set to the current axis position and the brake is deactivated, if necessary. Now the axis can move again and you can command a new target position.

***INFORMATION***

With the `CTO` (p. 164) and `TRO` (p. 238) commands, you can program the digital output lines of the C-863 so that they are enabled in case of motion errors. The programmed output lines remain enabled until the error code is reset to 0. Details see "Setting Up "Motion Error" Trigger Mode" (p. 102).

## 8.2 Data Recorder

### 8.2.1 Data Recorder Properties

The C-863 contains a real-time data recorder. The data recorder can record different values for the axis (e.g. current position) and analog input signals.

The recorded data is temporarily stored in 2 data recorder tables with 1024 points each. Each data recorder table contains the data of one data source.

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

### 8.2.2 Setting up the Data Recorder

***INFORMATION***

The settings for setting up the data recorder can only be changed in the volatile memory of the C-863. After the C-863 has been switched on or rebooted, factory settings will be active unless a configuration takes place with a start-up macro.

#### Reading out general information about the data recorder

➢ Send the `HDR?` command (p. 189).

  The available record options and trigger options as well as information on additional parameters and commands for data recording are displayed.

### Configuring the data recorder

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

- ➢ Send the `DRC?` command (p. 177) to read out the current configuration. Data recorder tables with the record option 0 are deactivated, i.e., nothing is recorded. In the default setting, the two data recorder tables of the C-863 record the following:
  - – Data recorder table 1: Commanded position of the axis
  - – Data recorder table 2: Current position of the axis
- ➢ Configure the data recorder with the `DRC` command (p. 175).

You can specify how the recording is to be triggered.

- ➢ Get the current trigger option with `DRT?` (p. 181).
- ➢ Change the trigger option with the `DRT` command (p. 180). The trigger option applies to all data recorder tables whose record option is not set to 0.

### Setting sampling interval

- ➢ Send the `RTR?` command (p. 220) to read out the record table rate of the data recorder.

  The record table rate indicates after how many servo cycles a data point each is recorded. The default value is 10 servo cycles. The cycle time of the C-863 is 50 µs.
- ➢ Change the record table rate with the `RTR` command. (p. 219)

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

## 8.2.3 Starting the Recording

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

Regardless of the trigger option set, the data recording is always triggered when a step response measurement is started with `STE` (p. 231).

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.

### 8.2.4 Reading Out Recorded Data

***INFORMATION***

Reading out the recorded data can take some time, depending on the number of data points.

The data can also be read out while data is being recorded.

➢ Read out the last recorded data with the `DRR?` command.

The data is output in the GCS array format (see the SM146E user manual on the product CD).

## 8.3 Digital Output Signals

The digital outputs of the C-863 are available at the **I/O** socket (p. 298).

➢ Get the number of the output lines available on the C-863 with the `TIO?` command (p. 236).

External devices can be triggered via the digital outputs of the C-863. 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. 121).

### 8.3.1 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 on which the trigger output is enabled with `TRO`. |
| `TRO` | TRO {<TrigOutID> <TrigMode>} | Enables or disables the trigger output conditions set with `CTO`. Default: Trigger output disabled. |

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. 99).<br>Optionally, start and stop values can be defined to limit triggering to one position range and one particular direction of motion (negative or positive). |
| 2 | On Target | The on-target state of the axis selected is output at the selected trigger output (p. 101). |
| 5 | Motion Error | The selected digital output line becomes active when a motion error occurs (p. 102). The line stays active until the error code is reset to 0 (by a query with `ERR?`). |
| 6 | In Motion | The selected digital output line is active as long as the selected axis is in motion (p. 102). |
| 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. Details see "Setting Up "Position + Offset" Trigger Mode" (p. 103). |
| 8 | Single Position | The selected digital output line is active when the axis position has reached or exceeded a given position (p. 105). |

In addition, the polarity (active high / active low) of the signal can be set at the digital output (p. 105).

***INFORMATION***

The settings for the configuration of the digital output lines can only be modified in the volatile memory of the C-863. After the C-863 has been switched on or rebooted, factory default settings are enabled, provided that a configuration has not been carried out already with a start-up macro.

### 8.3.2 Setting Up "Position Distance" Trigger Mode

The *Position Distance* trigger mode lends itself to 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 (50 µs).

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 enable 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 stage 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 enable 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 stage 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
```

## 8.3.3 Setting Up "On Target" Trigger Mode

In the *On Target* trigger mode the on-target state of the axis selected (p. 30) is output at the selected trigger 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 2`, where *2* specifies the *On Target* trigger mode.
2. If you want to enable 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
```

## 8.3.4 Setting Up "Motion Error" Trigger Mode

The *Motion Error* trigger mode lends itself to monitoring motions. 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 a query with `ERR?`).

**INFORMATION**

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

Further information see "Motion Error" (p. 93).

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 enable the conditions for trigger output, send `TRO <TrigOutID> 1`.

## 8.3.5 Setting Up "In Motion" Trigger Mode

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

The motion state can also be read with the `#5` (p. 155), `#4` (p. 154) and `SRG?` (p. 230) 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 enable the conditions for trigger output, send `TRO <TrigOutID> 1`.

**Example:**

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

➢ Send:

```
CTO 1 2 1
```

```
CTO 1 3 6
```

```
TRO 1 1
```

### 8.3.6 Setting Up "Position + Offset" Trigger Mode

The *Position+Offset* trigger mode lends itself to scanning applications. The first trigger pulse is output when the axis has reached a specified position (TriggerPosition). The next trigger pulses each are output 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 for which direction of motion trigger pulses are to be output.

The pulse width is one servo cycle (50 µs).

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 first trigger pulse output.
   - Send `CTO <TrigOutID> 9 Stop`, where *Stop* indicates the stop value.
2. If you want to enable 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. Then a pulse should be output on this line every time axis 1 has covered a distance of 0.1 µm in 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. Then a pulse should be output on this line every time axis B has covered a distance of 1 µm in 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-863.

## 8.3.7 Setting Up "Single Position" Trigger Mode

In the *Single Position* trigger mode, the selected digital output line is active when the axis position has reached or exceeded a given 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>) 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 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 enable 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
```

## 8.3.8 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>) that is 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`

## 8.4 Digital Input Signals

The digital inputs of the C-863 are available on the **I/O** socket (p. 298).

➢ Get the number of the input lines available on the C-863 with the `TIO?` command (p. 236).

➢ Get the state of the input lines with the `DIO?` command (p. 175).

Potential applications:

- Use in macros (p. 109). Details and examples of macros can be found in "Controller Macros" (p. 121).
- Use as switch signals (p. 109)

***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

### 8.4.1 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, in combination with the DIO? query command, the state of a digital input line to a variable. Use in macros to set local variables (p. 148). |
| `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. |

| Command | Syntax | Function |
|---|---|---|
| `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. 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 point switch. A digital input line can be used as the source of the reference switch signal instead of the reference point switch. |
| `JRC` | JRC <Jump> <CMD?> <OP> <Value> | Can only be used in macros. Triggers a relative jump of the macro execution pointer depending on the state of a digital input line when used in combination with the DIO? query command. |
| `MEX` | MEX <CMD?> <OP> <Value> | Can only be used in macros. Stops the macro execution depending on the state of a digital input line when used in combination 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 combination with the DIO? query command. |

## Parameters

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

| Parameter | Description and possible values |
|---|---|
| ***Source Of Reference Signal***<br>0x5C | Specifies the source of the reference signal for the `FRF` and `FED` commands:<br>0 = reference point switch<br>1 = digital input 1<br>2 = digital input 2<br>3 = digital input 3<br>4 = digital input 4 |

| Parameter | Description and possible values |
|---|---|
| ***Source Of Negative Limit Signal***<br>0x5D | Specifies the source(s) of the negative limit switch signal for the `FNL` and `FED` commands via a bitmask:<br>0 = Negative limit switch (default setting)<br>1 = Digital input 1 (bit 0)<br>2 = Digital input 2 (bit 1)<br>4 = Digital input 3 (bit 2)<br>8 = Digital input 4 (bit 3) |
| ***Source Of Positive Limit Signal***<br>0x5E | Specifies the source(s) of the positive limit switch signal for the `FPL` and `FED` commands via a bitmask:<br>0 = Positive limit switch (default setting)<br>1 = Digital input 1 (bit 0)<br>2 = Digital input 2 (bit 1)<br>4 = Digital input 3 (bit 2)<br>8 = Digital input 4 (bit 3) |
| ***Invert Digital Input Used For Negative Limit***<br>0x5F | Inverts the polarity of the digital inputs that are used for the source of the negative limit switch signal via a bitmask:<br>0 = No digital input inverted (default setting).<br>1 = Digital input 1 inverted (bit 0)<br>2 = Digital input 2 inverted (bit 1)<br>4 = Digital input 3 inverted (bit 2)<br>8 = Digital input 4 inverted (bit 3) |
| ***Invert Digital Input Used For Positive Limit***<br>0x60 | Inverts the polarity of the digital inputs that are used for the source of the positive limit switch signal via a bitmask:<br>0 = No digital input inverted (default setting).<br>1 = Digital input 1 inverted (bit 0)<br>2 = Digital input 2 inverted (bit 1)<br>4 = Digital input 3 inverted (bit 2)<br>8 = Digital input 4 inverted (bit 3) |

## 8.4.2 Using Digital Input Signals in Macros

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

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

Further information and examples can be found in "Controller Macros" (p. 121).

***INFORMATION***

You can connect the C-170.PB pushbutton box from PI to the **I/O** socket (p. 298) to generate the digital input signals for use in macros. It also displays via LEDs the state of the digital output lines.

## 8.4.3 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 point 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 bearing 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 about changing parameters can be found in "Adapting Settings" (p. 263).

## 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-863 evaluates the signals from the internal limit switches of the stage. This parameter has no bearing 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 about changing parameters can be found in "Adapting Settings" (p. 263).

**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 only in the volatile memory of the C-863.

- ➢ 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.

## 8.5 Analog Input Signals

The analog inputs of the C-863 are available on the **I/O** socket (p. 298).

➢ Get the number of analog input lines available on the C-863 with the `TAC?` command (p. 235).

➢ Query the voltage on the analog inputs with the `TAV?` command (p. 235).

➢ Use the data recorder (p. 95) to record the analog input signals.

Potential applications:

- Use in macros (p. 112): Details and examples of macros are found in "Controller Macros" (p. 121).
- 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

### 8.5.1 Commands for Analog Inputs

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

| Command | Syntax | Function |
|---|---|---|
| `CPY` | CPY <Variable> <CMD?> | Copies, in combination with the TAV? query command, the voltage value of an analog input line to a variable. Use in macros to set local variables (p. 148). |
| `DRC` | DRC {<RecTableID> <Source> <RecOption>} | Configures the data recorder. Analog input values can be recorded using record option 81. |
| `JRC` | JRC <Jump> <CMD?> <OP> <Value> | Can only be used in macros. Triggers a relative jump of the macro execution pointer depending on the voltage at an analog input line when used in combination with the TAV? query command. |

| Command | Syntax | Function |
|---|---|---|
| `MEX` | MEX <CMD?> <OP> <Value> | Can only be used in macros. Stops macro execution depending on the voltage at an analog input line when used in combination with the TAV? query command. |
| `TAC?` | TAC? | Get the number of installed analog lines. |
| `TAV?` | TAV? [{<AnalogInputID>}] | Get voltage at analog input. |
| `WAC` | WAC <CMD?> <OP> <Value> | Can only be used in macros. Waits until an analog input line reaches a certain voltage when used in combination with the TAV? query command. |

### 8.5.2 Using Analog Input Signals in Macros

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

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

Further information and examples can be found in "Controller Macros" (p. 121).

## 8.6 Joystick Control

### 8.6.1 Functionality of the Joystick Control

The joystick axis controls the velocity of the stage axis connected to the C-863 (commanded velocity output from the profile generator).

The interrelationship between the displacement of the joystick axis and the velocity of the stage axis is established by the C-863 by the use of a lookup table. The 256 values in the lookup table are factors that are applied to the velocity set with the `VEL` command (p. 242) 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-863.

During the joystick control, the soft limit which is specified by the parameter 0x15 or 0x30 is used as the target position. Details see "Travel Range and Soft Limits" (p. 33). 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 enabled for the axis.

Joystick control is not possible in open-loop operation (servo mode OFF).

## 8.6.2 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>} | Enables or disables a joystick device which is 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 |
|---|---|
| ***Invert Direction Of Motion For Joystick-Controlled Axis?*** 0x61 | Specifies the direction of motion for joystick-controlled axes. 0 = direction of motion not inverted (default setting) 1 = direction of motion inverted |

## 8.6.3 Controlling Axis Motion

In the following, PIMikroMove is used to control the stage via a joystick. Here you do not have to deal with the respective GCS commands.

*NOTICE*

**Unintentional motions while enabling joystick!**

If no joystick is connected to the C-863, 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-863.

***INFORMATION***

The stage connected to the C-863 can only be controlled by one joystick axis.

***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. 121).

***INFORMATION***

The C-863 supports one logical axis and therefore is normally used with stages that only have one motion axis. In this case, the designation "axis" is synonymous with "stage". Therefore, no distinction is made between "stage" and "axis" in the following operational instructions.

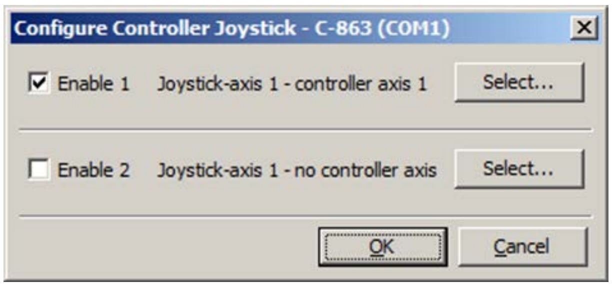
## Prerequisite

✓ You have started up the stage with PIMikroMove and started initial motions in closed-loop operation (p. 81).

✓ You have connected a joystick to the C-863 (p. 63).

## Controlling axis motion via a joystick

1. In the main window of PIMikroMove, open the ***Configure Controller Joystick*** window via the ***C-863 > Configure controller joystick(s)...*** menu item.



Configure Controller Joystick - C-863 (COM1)

☑ Enable 1 Joystick-axis 1 - controller axis 1

☐ Enable 2 Joystick-axis 1 - no controller axis

*Figure 28: Configure Controller Joystick*

The ***Configure Controller Joystick*** window lists the joystick devices supported by the C-863 and their joystick axes.

– Row 1 belongs to the joystick device that can be connected to pin 4 of the **Joystick** socket (e.g. C-819.20 or C-819.30 joystick from PI). The C-863 supports one joystick axis of this device.

– Row 2 belongs to the joystick device that can be connected to pin 2 of the **Joystick** socket. The C-863 supports one joystick axis of this device.

2. Assign the axis to be moved to a joystick axis of the joystick device connected:

a) In the ***Configure Controller Joystick*** window, click on the corresponding ***Select*** button.

b) In the ***Select controller axis*** window, select the correct stage name.

c) In the ***Select controller axis*** window, click on ***OK*** to confirm the selection and to close the window.

3. In the ***Configure Controller Joystick*** window, enable the joystick device connected by checking the respective ***Enable*** check box.

   If the stage moves even though you are not operating the joystick:

   – Check whether the joystick is locked mechanically.

   – Calibrate the joystick (p. 116).

4. Control the velocity of the stage via the joystick.

5. If you want to disable joystick control, remove the check from the respective ***Enable*** check box in the ***Configure Controller Joystick*** window.

## 8.6.4 Calibrating the Joystick

Calibration of the individual joystick axes is recommended after connecting a joystick to the C-863 for the first time. Calibration involves the following steps:

- If applicable operating elements are present on the joystick device: mechanical adjustment of the joystick axes.
- Calibration of joystick axes in PIMikroMove

***INFORMATION***

With joystick calibration in PIMikroMove, the lookup table type to be used is selected or the lookup table is filled with custom values. To do this, no stage needs to be connected to the C-863.

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

- Once joystick control is enabled, the stage moves even though you are not operating 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 non-volatile 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. 291) if the C-863 shows unexpected behavior.

***INFORMATION***

For the Z axis of the C-819.30 joystick, mechanical adjustment is not possible and it cannot be operated using the standard lookup table types (linear or parabolic) of the C-863.

- Calibrate the Z axis of the joystick after connecting to the C-863 with PIMikroMove.
- Use the ***Measure Joystick Parameters and Use Custom Lookup Table*** method for calibrating the Z axis.
- Repeat the calibration of the Z axis if you are connecting the Z axis to another controller.

***INFORMATION***

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

## Adjusting a joystick axis mechanically

➢ With joystick control enabled, check whether the stage is already moving in case you are not operating the joystick.

If so:

➢ Check whether the joystick is locked mechanically and disengage the lock if necessary.

➢ Keep the joystick axis in question (i. e. e. g. the control lever) in center position and adjust it with the appropriate operating elements until the stage is no longer moving. With the C-819.20 and C-819.30 joysticks, turn the corresponding rotary knob for adjustment (p. 119).

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 a joystick axis in PIMikroMove

1. In the main window of PIMikroMove, open the ***Joystick Calibration*** window via the ***C-863 > Calibrate controller joystick… > Joystick n*** menu item, where *n* indicates the joystick device connected.

   *n* = 1: joystick device that can be connected to pin 4 of the **Joystick** socket (e.g. C-819.20 or C-819.30 joystick from PI).

   *n* = 2: joystick device that can be connected to pin 2 of the **Joystick** socket.

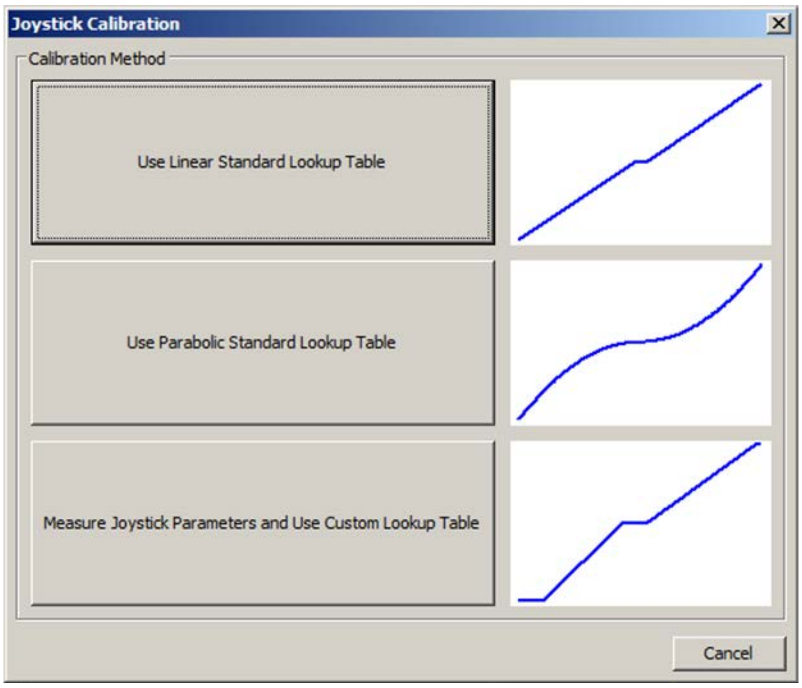

Joystick Calibration
Calibration Method
Use Linear Standard Lookup Table
Use Parabolic Standard Lookup Table
Measure Joystick Parameters and Use Custom Lookup Table
Cancel

*Figure 29: Joystick Calibration window for the calibration of a joystick axis*

   The ***Joystick Calibration*** window allows the calibration of the supported joystick axes of the joystick device connected – with C-863 one joystick axis each.

2. Select the preferred calibration method by clicking on the appropriate button:

   – If you want to use the linear lookup table type for the joystick axis, click on ***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 on ***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 show the behavior of the joystick in a custom lookup table, click on ***Measure Joystick Parameters and Use Custom Lookup Table***. Here the ***Controller Joystick Calibration*** window opens.

3. Once the ***Controller Joystick Calibration*** window has opened, follow the instructions in this window.

   In this way, custom lookup table values are determined.

   By clicking on ***OK*** you load the lookup table values determined to the non-volatile memory of the C-863. In doing so, the calibration is finished.

## 8.6.5 Joystick Devices Available

PI provides the joystick devices described in the following as optional accessories (p. 16).

### Analog C-819.20 joystick, 2 axes

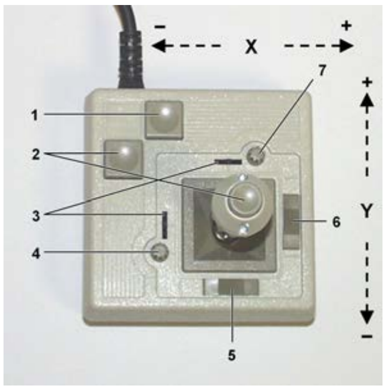


*Figure 30: C-819.20 joystick*

| | |
|---|---|
| *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

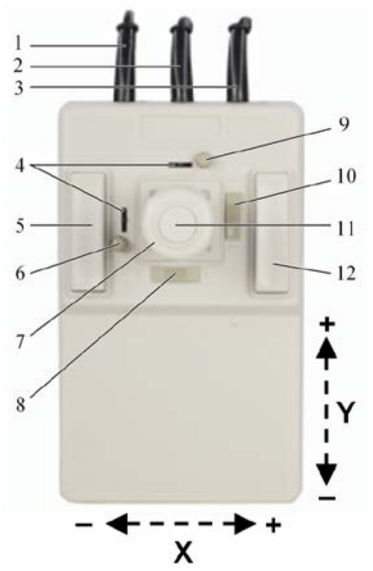


*Figure 31: C-819.30 joystick*

| | |
|---|---|
| *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* |

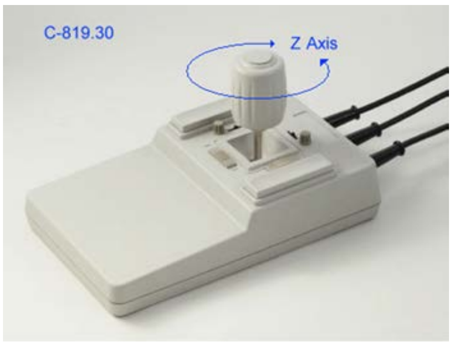


*Figure 32: C-819.30 joystick, rotary knob for the Z axis*

## 8.7 Controller Macros

### 8.7.1 Overview: Macro Functionalities and Example Macros

The C-863 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 start-up macro. The start-up macro is executed each time that the C-863 is switched on or rebooted.
- Processing a macro and stopping macro execution can be linked to conditions. In this way, loops can be realized as well.
- Macros can call up themselves or other macros in several nesting levels.
- Variables (p. 148) 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.

In this manual, you will find example macros for the following tasks:

- Moving an axis back and forth (p. 126)
- Recording a macro for a controller whose address is different from 1 (p. 127)
- Moving an axis with variable travel distance back and forth (p. 129)
- Implementing multiple calls of a macro via a loop (p. 130)
- Preparing an axis via start-up macro for closed-loop operation (p. 132)
- Synchronization of two controllers (p. 133)
- Stopping motion by pushbutton (p. 135)
- Joystick control with storage of positions (p. 136)
- Joystick control with change in velocity (p. 141)

## 8.7.2 Commands and Parameters for Macros

### Commands

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

| Command | Syntax | Function |
|---|---|---|
| `ADD` (p. 158) | ADD <Variable> <FLOAT1> <FLOAT2> | Adds two values and saves the result to a variable (p. 148). Can only be used for local variables in macros. |
| `CPY` (p. 163) | CPY <Variable> <CMD?> | Copies a command response to a variable (p. 148). Can only be used for local variables in macros. |
| `DEL` (p. 171) | DEL <uint> | Can only be used in macros. Delays <uint> milliseconds. |
| `JRC` (p. 202) | 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. 205) | MAC BEG <macroname> | Starts the recording of a macro with the name *macroname* on the controller. *macroname* can consist of up to 8 characters. |
| | MAC DEF <macroname> | Defines the given macro as the start-up macro. |
| | MAC DEF? | Gets the start-up macro. |
| | MAC DEL <macroname> | Deletes the given macro. |
| | MAC END | Ends the macro recording. |
| | MAC ERR? | Reports the last error which occurred during macro execution. |
| | MAC NSTART <macroname> <uint> [<String1> [<String2>]] | Starts the given macro n times in succession (n = number of executions). The values of local variables can be set for the macro with <String1> and <String2>. |
| | MAC START <macroname> [<String1> [<String2>]] | Starts one execution of the specified macro. The values of local variables can be set for the macro with <String1> and <String2>. |

| Command | Syntax | Function |
|---|---|---|
| MAC? (p. 208) | MAC? [<macroname>] | Lists all macros or the content of a given macro. |
| MEX (p. 209) | MEX <CMD?> <OP> <Value> | Can only be used in macros. Stops the macro execution depending on a condition. |
| RMC? (p. 216) | RMC? | Lists macros which are currently running. |
| VAR (p. 240) | VAR <Variable> <String> | Sets a variable (p. 148) to a certain value or deletes it. Can only be used for local variables in macros. |
| VAR? (p. 242) | VAR ? [{<Variable>}] | Gets variable values. |
| WAC (p. 244) | WAC <CMD?> <OP> <Value> | Can only be used in macros. Waits until a condition is met. |
| #8 (p. 156) | - | Tests if a macro is running on the controller. |

### Parameter

The following parameter is available for working with macros:

| Parameter | Description and possible values |
|---|---|
| ***Ignore Macro Error?*** 0x72 | Determines whether the controller macro is stopped when an error occurs while it is being executed. ▪ 0 = Stop macro when error occurs (default) ▪ 1 = Ignore error |

## 8.7.3 Working with Macros

Work with macros comprises the following:

- Recording of macros (p. 124)
- Starting macro execution (p. 128)
- Stopping macro execution (p. 131)
- Setting up a start-up macro (p. 131)
- Deleting of macros (p. 133)

**INFORMATION**

For working with controller macros, it is recommended to use the ***Controller macros*** tab in PIMikroMove. There you can conveniently record, start and manage controller macros. Details are found in the PIMikroMove manual.

## Recording a macro

**INFORMATION**

The C-863 can save up to 32 macros simultaneously. A maximum of 5 nesting levels are possible in macros.

**INFORMATION**

Basically all GCS commands (p. 143) can be included in a macro. Exceptions:

- `RBT` for rebooting the C-863
- `MAC BEG` and `MAC END` for macro recording
- `MAC DEL` for deleting a macro

Query commands can be used in macros in combination 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-863 whose controller address differs from 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. See “Variables” (p. 148) for more information.

**_INFORMATION_**

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

- ➢ Only record macros when it is necessary.
- ➢ Use variables (p. 148) in macros to make macros more flexible, and give the corresponding variable values when starting macro execution.
- ➢ Contact our customer service department (p. 291) if the C-863 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 ofPIMikroMove: Send the `MAC BEG macroname` command, where *macroname* 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 *macroname* 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 ofPIMikroMove: 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-863.
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 ofPIMikroMove:

– Get which macros are saved in the C-863 by sending the `MAC?` command.

– Get the contents of the *macroname* 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 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 omitted.

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

➢ Record the macros by sending:

```
MAC BEG macro1

MVR 1 12.5

WAC ONT? 1 = 1

MAC END

MAC BEG macro2

MVR 1 -12.5

WAC ONT? 1 = 1

MAC END

MAC BEG macro3
MAC START macro1
MAC START macro2
MAC END
```

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

***INFORMATION***

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

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 in every command line).

The servo mode is to be switched on for axis 1 via the ref macro and a reference move to the reference point switch is to be 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 execution

***INFORMATION***

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

***INFORMATION***

Simultaneous execution of multiple macros is not possible. Only one macro can be executed 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 found in the PIMikroMove manual.

1. If the macro execution is to be continued despite the occurrence of 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. 263).
2. Start the macro execution:
   - If the macro is to be executed once, send the `MAC START macroname string` command, whereby *macroname* indicates the name of the macro.
   - If the macro is to be executed n times, send the `MAC NSTART macroname n string` command, whereby *macroname* indicates the name of the macro and *n* indicates the number of executions.

*string* stands for the values of local variables. The values only have to be given 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 the local variable 1. The individual values must be separated from each other with spaces.

3. If you want to check the macro execution:
   - Get whether a macro is being executed on the controller by sending the `#8` command.
   - Get the name of the macro that is currently being executed 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 omitted.

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 thus 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 that the C-863 is switched on or rebooted, since they are only written to the volatile memory of the C-863.

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 given by the value of the local variable 1 and waits until the axis has reached the target position.

4. Start the execution of 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 omitted.

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 execution

***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 found in the PIMikroMove manual.

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

## Setting up a start-up macro

Any macro can be defined as the start-up macro. The start-up macro is executed each time that the C-863 is switched on or rebooted.

***INFORMATION***

Deleting a macro does not delete its selection as the start-up 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 found in the PIMikroMove manual.

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

**Example: Preparing an axis via a start-up macro for closed-loop operation**

***INFORMATION***

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

The STARTCL macro for axis 1 deactivates joystick control, switches on the servo mode and starts a reference move to the negative limit switch. As STARTCL is defined as the start-up macro, axis 1 is ready for closed-loop 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 using this macro, the parameter settings of the C-863 should be adapted in the non-volatile memory to the stage connected. Alternatively, the parameter settings can also be configured in the volatile memory via the start-up macro. Further information see "Adapting Settings" (p. 263).

### Deleting a macro

***INFORMATION***

A running macro may not be deleted.

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 found in the PIMikroMove manual.

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

## 8.7.4 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 omitted.

| 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. | -<br>Use a suitable cable. Pin assignment see "I/O" (p. 298). | 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. |

| Action | Command | Result |
|---|---|---|
| Set up the motion on the master controller and on the slave controller. | `SVO 1 1`<br>`FRF 1 1`<br>`VEL 1 0`<br>`MOV 1 5.5` | For both controllers: The servo mode is switched on and the axis has executed a reference move – here to the reference point 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. |
| Record the MASTER macro on the master controller. | `MAC BEG master`<br>`DIO 1 1`<br>`VEL 1 100`<br>`MAC END` | The macro has the following tasks:<br>■ Switch the digital output line 1 of the master controller to high state to trigger the slave controller<br>■ Set velocity to 100 to start the motion |
| Record the SLAVE macro on the slave controller. | `MAC BEG slave`<br>`WAC DIO? 1 = 1`<br>`VEL 1 100`<br>`MAC END` | The macro has the following tasks:<br>■ Set condition: The macro is executed further only if digital input line 1 has the high state (i.e. if the master controller outputs the trigger signal).<br>■ Set velocity to 100 to start the motion |
| 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 further macro execution 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. |

## 8.7.5 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 via LEDs the state of the digital output lines.

***INFORMATION***

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

| Action | Command | Result |
|---|---|---|
| Connect digital input line 1 on the **I/O** socket to an appropriate signal source. | -<br>Pin assignment see "I/O" (p. 298). | 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`<br>`MVR 1 5`<br>`JRC 2 DIO? 1 = 1`<br>`JRC -1 ONT? 1 = 0`<br>`HLT 1`<br>`MAC END` | The macro has the following tasks:<br>▪ Start relative motion of axis 1<br>▪ Set 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.<br>▪ Set 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. |
| 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. |

| Action | Command | Result |
|---|---|---|
| If error code 10 interferes: Record alternative HALTVAR macro which uses a variable. Details see "Variables" (p. 148). | `MAC BEG haltvar`<br>`MVR 1 5`<br>`JRC 2 DIO? 1 = 1`<br>`JRC -1 ONT? 1 = 0`<br>`CPY TARGET POS? 1`<br>`MOV 1 ${TARGET}`<br>`VAR TARGET`<br>`MAC END` | The macro has the same tasks as the HALT macro. However with pushbutton, axis 1 is not stopped 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. |

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

**Task:**

Axis 1 is to be controlled with a joystick. Joystick control is to be enabled only when the joystick button is pressed simultaneously. By using the buttons of a connected pushbutton box, in addition up to four positions are to be stored in the controller or approached to by the axis. The LEDs of the pushbutton box shall indicate whether the controller is ready to store the current position and whether storage is effected.

**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 omitted.

| Action | Command | Result |
|---|---|---|
| Connect C-170.PB pushbutton box from PI to the **I/O** 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 **Joystick** socket. | - | For commands, the joystick axis connected is accessible as axis 1 of joystick device 1 and the joystick button is accessible as button 1 of joystick device 1. |
| Switch on servo mode for axis 1. | `SVO 1 1` | The servo mode must be switched on, 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 point 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 device 1. Joystick control is still not enabled. |
| Record the STARTUP macro on the controller. | `MAC BEG startup`<br>`CPY STORE1 POS? 1`<br>`CPY STORE2 POS? 1`<br>`CPY STORE3 POS? 1`<br>`CPY STORE4 POS? 1`<br>`MAC START MAINLOOP`<br>`MAC END` | The macro has the following tasks:<br>▪ Initialize variables for storing the position<br>▪ Start MAINLOOP macro for the main loop |

| Action | Command | Result |
|---|---|---|
| Record the MAINLOOP macro on the controller. | `MAC BEG mainloop`<br>`MAC START TESTJOYB`<br>`VAR COUNTER 1`<br>`MAC START TESTDION ${COUNTER}`<br>`ADD COUNTER ${COUNTER} 1`<br>`JRC -2 VAR? COUNTER < 5`<br>`MAC START MAINLOOP`<br>`MAC END` | The macro has the following tasks:<br>▪ Start TESTJOYB macro for joystick control<br>▪ Start TESTDION macro successively for all digital inputs (i.e. every pushbutton box button), using a loop<br>▪ Call itself to set up the main loop |
| Record the TESTJOYB macro on the controller. | `MAC BEG testjoyb`<br>`MEX JBS? 1 1 = 0`<br>`JON 1 1`<br>`DIO 0 15`<br>`JRC 6 JBS? 1 1 = 0`<br>`DEL 50`<br>`DIO 0 0`<br>`JRC 3 JBS? 1 1 = 0`<br>`DEL 50`<br>`JRC -6 JBS? 1 1 = 1`<br>`JON 1 0`<br>`DIO 0 0`<br>`MAC END` | The macro has the following tasks:<br>▪ Stop macro execution if joystick button 1 is not pressed<br>▪ Enable joystick device 1<br>▪ Switch all LEDs on the pushbutton box on<br>▪ Jump forward 6 lines (to JON 1 0), if joystick button 1 is no longer pressed<br>▪ Wait 50 ms<br>▪ Switch all LEDs on the pushbutton box off<br>▪ Jump forward 3 lines (to JON 1 0), if joystick button 1 is no longer pressed<br>▪ Wait additional 50 ms<br>▪ Jump back 6 lines (to DIO 0 15), if joystick button 1 is still pressed<br>▪ Disable joystick device 1<br>▪ Switch all LEDs on the pushbutton box off |

| Action | Command | Result |
|---|---|---|
| Record the TESTDION macro on the controller. | `MAC BEG testdion` | The macro has the following tasks: |
| | `MEX VAR? 0 != 1` | ▪ Stop macro execution if the number of local variables given when starting TESTDION is not 1 |
| | `MEX DIO? $1 = 0` | ▪ Stop macro execution if the pushbutton box button specified via local variable 1 is no longer pressed (corresponding input line has the low state) |
| | `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 macro execution if button is no longer pressed |
| | `DEL 400` | ▪ Wait 400 ms |
| | `MEX DIO? $1 = 0` | ▪ Stop macro execution if button is no longer pressed |
| | `DIO $1 1` | ▪ Activate the pushbutton box LED associated with the button pressed to indicate the storage of the current position |
| | `WAC DIO? $1 = 0` | ▪ The macro is executed further only if the button is no longer pressed |
| | `DIO $1 0` | ▪ Deactivate LED |

| Action | Command | Result |
|---|---|---|
| | `CPY STORE$1 POS? 1`<br><br>`MAC END` | ▪ Store the current position of axis 1 in the global variable designated via local variable 1 |
| Record the MVAX2ST macro on the controller. | `MAC BEG MVAX2ST`<br>`CPY 2 VAR? STORE$1`<br><br>`MOV 1 $2`<br><br>`MAC END` | The macro has the following tasks:<br>▪ Get the storage variable designated via local variable 1 and copies its value to local variable 2<br>▪ Start motion of axis 1 to the target position specified via local variable 2 |
| Start the STARTUP macro on the controller.<br>Alternative:<br>If the variables for storing positions are not to be initialized, start the MAINLOOP macro on the controller instead. | `MAC START startup` | Joystick control is enabled by pressing the joystick button. When joystick control is enabled, the pushbutton box LEDs flash rapidly and thereby indicate that the box buttons should not be pressed. After releasing the joystick button, joystick control is disabled and the LEDs switch themselves off. The pushbutton box can now be used for approaching the stored positions or for storing the current position.<br>To move the stage to a stored position, the respective pushbutton box button is pressed briefly.<br>To store the current position of the stage, one button on the pushbutton box is pressed until the button LED lights up. |

### 8.7.7 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 omitted.

| Action | Command | Result |
|---|---|---|
| Connect C-819.20 or C-819.30 joystick to the **Joystick** socket. | - | For commands, the joystick axis connected is accessible as axis 1 of joystick device 1 and the joystick button is accessible as button 1 of joystick device 1. |
| Switch on servo mode for axis 1. | `SVO 1 1` | The servo mode must be switched on, 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 point 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 device 1. Joystick control is still not enabled. |
| Record the JOYVEL macro on the controller. | `MAC BEG joyvel`<br>`JON 1 1`<br>`JRC 3 JBS? 1 1 = 1`<br>`VEL 1 0.5`<br>`JRC -2 JBS? 1 1 = 0`<br>`VEL 1 1`<br>`JRC -4 JON? 1 = 1`<br>`MAC END` | The macro has the following tasks:<br>▪ Enable joystick device 1<br>▪ If joystick button 1 is pressed, jump 3 lines forward (to `VEL 1 1`).<br>▪ Maximum velocity during joystick control is 0.5.<br>▪ If joystick button 1 is not pressed, jump 2 lines back to set up a loop.<br>▪ Maximum velocity during joystick control is 1.<br>▪ If joystick device 1 is still enabled, jump 4 lines back to set up a loop. |

| Action | Command | Result |
|---|---|---|
| Start the JOYVEL macro on the controller. | `MAC START joyvel` | Motion with low velocity:<br>Move the joystick control lever.<br>Motion with high velocity:<br>Keep the pushbutton 1 of the joystick pressed and move the control lever. |

# 9 GCS Commands

## In this Chapter

## 9.1 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 item identifier or a command-specific parameter |
| […] | Square brackets indicate an optional entry |
| {…} | Curly brackets indicate a repetition of entries, i.e. that it is possible to access more than one item (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. |

## 9.2 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. The response to a single-character commands is followed by a termination character, however.

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. See "Commandable Items" (p. 18) for the identifiers supported by the C-863.

Example 1:

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

Send: `MOVSP1SP10.0LF`

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

Example 2:

Two axes which are 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 omitted, 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 omitted in the last line:

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

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

In the response, the arguments are listed 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>SP LF

<Arg1>"="<Val1>SP LF

<Arg2>"="<Val2>LF

Example 5:

Send: `TSP?SP2SP1LF`

Receive:

```
2=-1158.4405SP LF
1=+0000.0000LF
```

***INFORMATION***

With the C-863 only a single element per command line can be addressed (e.g. axis, channel 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.

## 9.3 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. But because only the PC may send command lines to the controllers, its address (0) can be omitted. Both target and sender address, however, are part of any controller reply.

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 reply 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 omitted when addressing a controller, the target and sender addresses will also be omitted in the reply of the controller.

Example:

Send: `*IDN?`

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

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

Send: `1 *IDN?`

The same controller replies with:

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

See "Adapting DIP Switch Settings" for 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 replies are sent to the PC.

## 9.4 Variables

For more flexible programming, the C-863 supports variables. While global variables are always available, local variables are only valid for a given 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 must not contain special characters (especially no “$”).
- 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 must not contain alphabetic characters. Possible characters are 0 to 9.
- The variable name can also be given 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 curly brackets.

If the variable name consists of a single character, the curly brackets can be omitted.

Note that if the curly brackets are omitted with variable names consisting of multiple characters, the first character after the “$” is interpreted as the variable name.

**Local variables:**

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

  The command formats are:

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

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

  <STRING1> and <STRING2> indicate the values for the local variables 1 and 2 used in the macro. <STRING1> and <STRING2> can be given directly or via the values of variables. <uint> defines the number of times the macro is to be run. See the `MAC` command (p. 205) 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. 158), `CPY` (p. 163) or `VAR` (p. 240), 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?`.

## 9.5 Command Overview

| Command | Format | Description |
|---|---|---|
| #4 (p. 154) | #4 | Request Status Register |
| #5 (p. 155) | #5 | Request Motion Status |
| #7 (p. 155) | #7 | Request Controller Ready Status |
| #8 (p. 156) | #8 | Query If Macro Is Running |
| #24 (p. 156) | #24 | Stop All Axes |
| *IDN? (p. 157) | *IDN? | Get Device Identification |
| ACC (p. 157) | ACC {<AxisID> <Acceleration>} | Set Closed-Loop Acceleration |
| ACC? (p. 158) | ACC? [{<AxisID>}] | Get Closed-Loop Acceleration |
| ADD (p. 158) | ADD <Variable> <FLOAT1> <FLOAT2> | Add and Save To Variable |
| BRA (p. 161) | BRA {<AxisID> <BrakeState>} | Set Brake Activation State |
| BRA? (p. 162) | BRA? [{<AxisID>}] | Get Brake Activation State |

| Command | Format | Description |
|---|---|---|
| CPY (p. 163) | CPY <Variable> <CMD?> | Copy Into Variable |
| CST? (p. 164) | CST? [{<AxisID>}] | Get Assignment Of Stages To Axes |
| CSV? (p. 164) | CSV? | Get Current Syntax Version |
| CTO (p. 164) | CTO {<TrigOutID> <CTOPam> <Value>} | Set Configuration Of Trigger Output |
| CTO? (p. 169) | CTO? [{<TrigOutID> <CTOPam>}] | Get Configuration Of Trigger Output |
| DEC (p. 170) | DEC {<AxisID> <Deceleration>} | Set Closed-Loop Deceleration |
| DEC? (p. 170) | DEC? [{<AxisID>}] | Get Closed-Loop Deceleration |
| DEL (p. 171) | DEL <uint> | Delay The Command Interpreter |
| DFH (p. 171) | DFH [{<AxisID>}] | Define Home Position |
| DFH? (p. 173) | DFH? [{<AxisID>}] | Get Home Position Definition |
| DIO (p. 174) | DIO {<DIOID> <OutputOn>} | Set Digital Output Lines |
| DIO? (p. 175) | DIO? [{<DIOID>}] | Get Digital Input Lines |
| DRC (p. 175) | DRC {<RecTableID> <Source> <RecOption>} | Set Data Recorder Configuration |
| DRC? (p. 177) | DRC? [{<RecTableID>}] | Get Data Recorder Configuration |
| DRR? (p. 178) | DRR? [<StartPoint> <NumberOfPoints> [{<RecTableID>}]] | Get Recorded Data Values |
| DRT (p. 180) | DRT {<RecTableID> <TriggerSource> <Value>} | Set Data Recorder Trigger Source |
| DRT? (p. 181) | DRT? [{<RecTableID>}] | Get Data Recorder Trigger Source |
| ERR? (p. 182) | ERR? | Get Error Number |
| FED (p. 182) | FED {<AxisID> <EdgeID> <Param>} | Find Edge |
| FNL (p. 184) | FNL [{<AxisID>}] | Fast Reference Move To Negative Limit |
| FPL (p. 185) | FPL [{<AxisID>}] | Fast Reference Move To Positive Limit |
| FRF (p. 187) | FRF [{<AxisID>}] | Fast Reference Move To Reference Switch |
| FRF? (p. 188) | FRF? [{<AxisID>}] | Get Referencing Result |
| GOH (p. 189) | GOH [{<AxisID>}] | Go To Home Position |

| Command | Format | Description |
|---|---|---|
| HDR? (p. 189) | HDR? | Get All Data Recorder Options |
| HLP? (p. 191) | HLP? | Get List of Available Commands |
| HLT (p. 191) | HLT [{<AxisID>}] | Halt Motion Smoothly |
| HPA? (p. 192) | HPA? | Get List Of Available Parameters |
| JAS? (p. 193) | JAS? [{<JoystickID> <JoystickAxis>}] | Query Joystick Axis Status |
| JAX (p. 194) | JAX <JoystickID> <JoystickAxis> <AxisID> | Set Axis Controlled By Joystick |
| JAX? (p. 195) | JAX? [{<JoystickID> <JoystickAxis>}] | Get Axis Controlled By Joystick |
| JBS? (p. 196) | JBS? [{<JoystickID> <JoystickButton>}] | Query Joystick Button Status |
| JDT (p. 196) | JDT {<JoystickID> <JoystickAxis> <uint>} | Set Joystick Default Lookup Table |
| JLT (p. 197) | JLT <JoystickID> <JoystickAxis> <Addr> <floatn> | Fill Joystick Lookup Table |
| JLT? (p. 199) | JLT? [<StartPoint> <NumberOfPoints> [{<JoystickID> <JoystickAxis>}]] | Get Joystick Lookup Table Values |
| JON (p. 201) | JON {<JoystickID> <uint>} | Set Joystick Activation Status |
| JON? (p. 202) | JON? [{<JoystickID>}] | Get Joystick Activation Status |
| JRC (p. 202) | JRC <Jump> <CMD?> <OP> <Value> | Jump Relatively Depending On Condition |
| LIM? (p. 204) | LIM? [{<AxisID>}] | Indicate Limit Switches |
| MAC (p. 205) | MAC <keyword> {<parameter>}<br>in particular:<br>MAC BEG <macroname><br>MAC DEF <macroname><br>MAC DEF?<br>MAC DEL <macroname><br>MAC END<br>MAC ERR?<br>MAC NSTART <macroname> <uint> [<String1> [<String2>]]<br>MAC START <macroname> [<String1> [<String2>]] | Call Macro Function |
| MAC? (p. 208) | MAC? [<macroname>] | List Macros |

| Command | Format | Description |
|---|---|---|
| MEX (p. 209) | MEX <CMD?> <OP> <Value> | Stop Macro Execution Due To Condition |
| MOV (p. 211) | MOV {<AxisID> <Position>} | Set Target Position |
| MOV? (p. 212) | MOV? [{<AxisID>}] | Get Target Position |
| MVR (p. 213) | MVR {<AxisID> <Distance>} | Set Target Relative To Current Position |
| ONT? (p. 214) | ONT? [{<AxisID>}] | Get On-Target State |
| POS (p. 215) | POS {<AxisID> <Position>} | Set Real Position |
| POS? (p. 216) | POS? [{<AxisID>}] | Get Real Position |
| RBT (p. 216) | RBT | Reboot System |
| RMC? (p. 216) | RMC? | List Running Macros |
| RON (p. 217) | RON {<AxisID> <ReferenceOn>} | Set Reference Mode |
| RON? (p. 218) | RON? [{<AxisID>}] | Get Reference Mode |
| RPA (p. 218) | RPA [{<ItemID> <PamID>}] | Reset Volatile Memory Parameters |
| RTR (p. 219) | RTR <RecordTableRate> | Set Record Table Rate |
| RTR? (p. 220) | RTR? | Get Record Table Rate |
| SAI (p. 220) | SAI {<AxisID> <NewIdentifier>} | Set Current Axis Identifiers |
| SAI? (p. 221) | SAI? [ALL] | Get List Of Current Axis Identifiers |
| SEP (p. 222) | SEP <Pswd> {<ItemID> <PamID> <PamValue>} | Set Non-Volatile Memory Parameters |
| SEP? (p. 223) | SEP? [{<ItemID> <PamID>}] | Get Non-Volatile Memory Parameters |
| SMO (p. 224) | SMO {<AxisID> <ControlValue>} | Set Open-Loop Control Value |
| SMO? (p. 225) | SMO? [{<AxisID>}] | Get Control Value |
| SPA (p. 226) | SPA {<ItemID> <PamID> <PamValue>} | Set Volatile Memory Parameters |
| SPA? (p. 229) | SPA? [{<ItemID> <PamID>}] | Get Volatile Memory Parameters |
| SRG? (p. 230) | SRG? {<AxisID> <RegisterID>} | Query Status Register Value |

| Command | Format | Description |
|---|---|---|
| STE (p. 231) | STE <AxisID> <Amplitude> | Start Step And Response Measurement |
| STP (p. 232) | STP | Stop All Axes |
| SVO (p. 233) | SVO {<AxisID> <ServoState>} | Set Servo Mode |
| SVO? (p. 235) | SVO? [{<AxisID>}] | Get Servo Mode |
| TAC? (p. 235) | TAC? | Tell Analog Channels |
| TAV? (p. 235) | TAV? [{<AnalogInputID>}] | Get Analog Input Voltage |
| TIO? (p. 236) | TIO? | Tell Digital I/O Lines |
| TMN? (p. 237) | TMN? [{<AxisID>}] | Get Minimum Commandable Position |
| TMX? (p. 237) | TMX? [{<AxisID>}] | Get Maximum Commandable Position |
| TNR? (p. 238) | TNR? | Get Number Of Record Tables |
| TRO (p. 238) | TRO {<TrigOutID> <TrigMode>} | Set Trigger Output State |
| TRO? (p. 239) | TRO? [{<TrigOutID>}] | Get Trigger Output State |
| TRS? (p. 239) | TRS? [{<AxisID>}] | Indicate Reference Switch |
| TVI? (p. 240) | TVI? | Tell Valid Character Set For Axis Identifiers |
| VAR (p. 240) | VAR <Variable> <String> | Set Variable Value |
| VAR? (p. 242) | VAR? [{<Variable>}] | Get Variable Value |
| VEL (p. 242) | VEL {<AxisID> <Velocity>} | Set Closed-Loop Velocity |
| VEL? (p. 243) | VEL? [{<AxisID>}] | Get Closed-Loop Velocity |
| VER? (p. 244) | VER? | Get Versions Of Firmware And Drivers |
| WAC (p. 244) | WAC <CMD?> <OP> <Value> | Wait For Condition |
| WPA (p. 245) | WPA <Pswd> [{<ItemID> <PamID>}] | Save Parameters To Non-Volatile Memory |

## 9.6 Command Descriptions for GCS 2.0

**#4 (Request Status Register)**

| | |
|---|---|
| Description: | Requests system status information. |
| Format: | #4 |
| Arguments: | None |
| Response: | The answer is bit-mapped. See below for the individual codes. |
| Notes: | This command is identical in function to SRG? (p. 230), but only one character is sent via the interface. Therefore #4 can also be used while the controller is performing time-consuming tasks.<br><br>For the C-863, the response is the sum of the following codes, in hexadecimal format: |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Descript-ion | On-target state | Deter-mines the reference value | In motion | Servo mode on | - | - | - | Error flag |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Descript-ion | Digital in-put line 4 | Digital in-put line 3 | Digital in-put line 2 | Digital in-put line 1 | - | Positive limit switch | Ref-erence point switch | Nega-tive limit switch |

Example: Send: `#4`

Receive: `0x9005`

Note: The response is given in hexadecimal format. It means that the axis is on target (on-target state =true), the servo mode is on, no error has occurred, the states of the digital input lines 1 to 4 are low, and the stage is on the positive side of the reference point 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 state of the axes. |
| Format: | #5 |
| Arguments: | None |
| Response: | The answer <uint> is bit-mapped and returned as the hexadecimal sum of the following codes:<br><br>1=First axis is moving<br>2=Second axis is moving<br>4=Third axis is moving<br>... |
| Examples: | 0 indicates motion of all axes complete<br>3 indicates that the first and the second axis are moving |

### #7 (Request Controller Ready Status)

| | |
|---|---|
| Description: | Asks controller for ready status (tests if controller is ready to perform a new command).<br><br>Note: Use #5 (p. 155) instead of #7 to verify if motion has finished. |
| Format: | #7 |
| Arguments: | None |
| Response: | B1h (ASCII character 177 = "±" in Windows) if controller is ready<br><br>B0h (ASCII character 176 = "°" in Windows) if controller is not ready<br>(e. g. performing 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<br><uint>=1 a macro is currently running |

**#24 (Stop All Axes)**

| | |
|---|---|
| Description: | Stops all axes abruptly. For details see the notes below.<br><br>Sets error code to 10.<br><br>This command is identical in function to STP (p. 232), 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. 211), MVR (p. 213), GOH (p. 189), STE (p. 231), SMO (p. 224)), commands for reference point definition (FNL (p. 184), FPL (p. 185), FRF (p. 187)) and macros (MAC (p. 205)). Also stops macro execution.<br><br>After the axes are stopped, their target positions are set to their current positions.<br><br>HLT (p. 191) in contrast to #24 stops motion with given deceleration with regard to system inertia. |

### *IDN? (Get Device Identification)

| | |
|---|---|
| Description: | Reports the device identity number. |
| Format: | *IDN? |
| Arguments: | None |
| Response: | Single-line text terminated with a termination character (line feed) with controller name, serial number, and firmware version |
| Notes: | For C-863, *IDN? responds something like: |

```
(c)2013 Physik Instrumente(PI) Karlsruhe,
C-863, 112048255,1.2.0.0
```

### ACC (Set Closed-Loop Acceleration)

| | |
|---|---|
| Description: | Sets acceleration of given axes.<br><br>The ACC setting only takes effect when the given axis is in closed-loop operation (servo mode ON).<br><br>ACC can be changed while the axis is moving. |
| Format: | ACC {<AxisID> <Acceleration>} |
| Arguments: | <AxisID> is one axis of the controller<br><br><Acceleration> is the acceleration value in physical units/s$^2$. |
| Response: | None |
| Troubleshooting: | Illegal axis identifiers |
| Notes: | The lowest possible value for <Acceleration> is 0.<br><br>ACC changes the value of the ***Closed-Loop Acceleration (Phys. Unit/s$^2$)*** parameter (ID0xB) in the volatile memory (can be saved as default with WPA (p. 245) and changed with SPA (p. 226) and SEP (p. 222)). |

The maximum value which can be set with the ACC command is given by the ***Maximum Closed-Loop Acceleration (Phys. Unit/s$^2$)*** parameter (ID 0x4A) (can be changed with SPA (p. 226) and SEP (p. 222)).

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

| | |
|---|---|
| Description: | Gets the acceleration value set with ACC for closed-loop operation.<br><br>If all arguments are omitted, gets the value of all axes set with ACC. |
| Format: | ACC? [{<AxisID>}] |
| Arguments: | <AxisID> is one axis of the controller |
| Response: | {<AxisID>"="<float> LF}<br><br>where<br><br><float> is the acceleration value for closed-loop operation set with ACC, in physical units/s$^2$. |

### ADD (Add and Save to Variable)

| | |
|---|---|
| Description: | Adds two values and saves the result to a variable (p. 148).<br><br>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.<br><br><FLOAT1> is the first summand.<br><br><FLOAT2> is the second summand.<br><br>For the summands, floating point numbers are expected. They can be given 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:<br><br>`ADD C $A $B` |
| Example 2: | The name of the variable to which the result is to be copied is given via the value of another variable: |

Send: `VAR?`

Receive:

```
A=468
B=123
3Z=WORKS
```

Send: `ADD A${3Z} $A $B`

Send: `VAR?`

Receive:

```
A=468
B=123
AWORKS=591
3Z=WORKS
```

Send: `ADD ${3Z} $A $B`

Send: `VAR?`

Receive:

```
A=468
B=123
AWORKS=591
WORKS=591
3Z=WORKS
```

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 given as arguments of the 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", perform 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 given axes. |
| Format: | BRA {<AxisID> <BrakeState>} |
| Arguments: | <AxisID> is one axis of the controller<br><br><BrakeState> can have the following values:<br>0 = Brake deactivated<br>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 with BRA only if the servo mode is switched off. Secure the stage against unintentional motions before you deactivate the brake with BRA!
- Setting the servo mode with SVO (p. 233) influences the activation state of the brake:
  - Switching on the servo mode deactivates the brake.
  - Switching off the servo mode activates the brake.
- If a motion error occurs (p. 93), the servo mode is switched off and the brake is activated.

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

| | |
|---|---|
| Description: | Gets brake activation state of given axes.<br><br>If all arguments are omitted, gets state of all axes. |
| Format: | BRA? [{<AxisID>}] |
| Arguments: | <AxisID> is one axis of the controller |
| Response: | {<AxisID>"="<BrakeState> LF}<br><br>where<br><br><BrakeState> is the current brake activation state of the axis:<br>0 = Brake deactivated<br>1 = Brake activated |
| Troubleshooting: | Illegal axis identifier |

### CPY (Copy Into Variable)

| | |
|---|---|
| Description: | Copies a command response to a variable (p. 148).<br><br>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.<br><br><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 given 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: | Gets the name of the stage type that is connected to the given axis. |
| Format: | CST? [{<AxisID>}] |
| Arguments: | <AxisID> is one axis of the controller |
| Response: | {<AxisID>"="<string> LF}<br><br>where<br><br><string> is the name of the stage type that is assigned to the axis. |
| Notes: | The stage 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 stage using SPA (p. 226) or SEP (p. 222). You can find details in the parameter overview (p. 272). |

### CSV? (Get Current Syntax Version)

| | |
|---|---|
| Description: | Gets 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 given digital output line. |
| Format: | CTO {<TrigOutID> <CTOPam> <Value>} |

| | |
|---|---|
| Arguments: | <TrigOutID> is one digital output line of the controller, see below for details.<br><br><CTOPam> is the CTO parameter ID in decimal format, see below for the available IDs.<br><br><Value> is the value to which the CTO parameter is set, see below. |
| Response: | None |
| Notes: | The trigger output conditions will become active when enabled with TRO (p. 238). Do not use DIO (p. 174) on digital output lines for which the trigger output is enabled with TRO.<br><br>The CTO settings are lost when you power down or reboot the C-863. An easy way to keep them is to save them to a macro. |
| Output lines and trigger conditions available: | <TrigOutID> corresponds to digital output lines Output 1 to Output 4, IDs = 1 to 4; see "I/O" (p. 298).<br><br><CTOPam> parameter IDs available for C-863:<br>1 = TriggerStep<br>2 = Axis<br>3 = TriggerMode<br>7 = Polarity<br>8 = StartThreshold<br>9 = StopThreshold<br>10 = TriggerPosition<br><br><Value> available for the appropriate <CTOPam> ID:<br><br>for TriggerStep: step size<br><br>for Axis: the identifier of the axis to be connected to the digital output line. Irrelevant for the MotionError trigger mode. |

for TriggerMode (default value is 0):

- 0 = PositionDistance;
  a trigger pulse is written whenever the axis has covered the TriggerStep distance (<CTOPam> ID 1). Optionally, values for StartThreshold and StopThreshold (<CTOPam> IDs 8 and 9) can be defined to enable the trigger output for a limited position range and a certain direction of motion only (negative or positive; Note: In case 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.
- 2 = OnTarget;
  the on-target status of the selected axis is transferred to the selected digital output line (this status can also be read with the ONT? command).
- 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).
- 6 = InMotion;
  the selected digital output line is active as long as the selected axis is in motion (the in-motion state can also be read with #4, #5 or the SRG? command).
- 7 = Position+Offset;
  the first trigger pulse is written when the axis has reached the position given by TriggerPosition (<CTOPam> ID 10). The next trigger pulses each are written when the axis position equals the sum of the last valid trigger position and the step size given by TriggerStep (<CTOPam> ID 1). Trigger output ends when the axis position exceeds the value given by StopThreshold (<CTOPam> ID 9). The sign of the TriggerStep value determines for which direction of motion trigger pulses are to be output. Trigger processing is done by the DSP of the C-863.

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 where the first trigger pulse is to be output in the Position+Offset trigger mode (default value is 0.0)

Application examples and further details see "Digital Output Signals" (p. 97) and the lines below.

| | |
|---|---|
| Example 1: | A pulse on the digital Output 1 line (ID 1) is to be generated whenever the axis 1 has covered a distance of 0.05 µm. The following parameters must be set:<br><br>TrigOutID = 1<br>Axis = 1<br>TriggerMode = 0<br>TriggerStep = 0.05<br>Send: `CTO 1 2 1`<br>Send: `CTO 1 3 0`<br>Send: `CTO 1 1 0.00005` |
| Example 2: | In this example, the digital output line 1 shall be set from low to high when axis A starts to move. The following parameters must be set:<br>TrigOutID = 1<br>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: M-122.2DD is connected to axis 1. The reference position of M-122.2DD is 18.5 mm. Starting from its reference position, the axis is to be moved alternating forwards and backwards; trigger pulses are to be output for both directions of motion in a range of 1 mm using the Position+Offset trigger mode. For that purpose, two macros are written to the controller. Macro TRIGREF initializes the controller and could also be defined as start-up macro, while macro TRIGGER starts motion and hence trigger output. Write the macros as shown below. Further details about macros see "Working with Macros" (p. 123).

Make sure that the velocity setting for the axis is suitable for the *Stepsize* setting made with CTO. Recommended value:
maximum velocity = *Stepsize* * 20 kHz / 2
where 20 kHz is the frequency of the C-863 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 20.5
CTO 1 10 19.5
```

```
DEL 1000
DRT 0 2 0
MOV 1 20.51
WAC POS? 1 > 20.3
MEX CTO? 1 10 < 19.4
CTO 1 1 -0.02
CTO 1 9 19.5
CTO 1 10 20.5
DEL 1000
MOV 1 19.49
WAC POS? 1 < 19.5
MEX CTO? 1 10 > 19.6
MAC START TRIGGER
```

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

| | |
|---|---|
| Description: | Gets 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.<br><br><CTOPam>: parameter ID; see CTO.<br><br>If all arguments are omitted, the response contains the values for all parameters and all output lines. |
| Response: | {<TrigOutID> <CTOPam>"="<Value> LF}<br><br>For <Value> see CTO. |

### DEC (Set Closed-Loop Deceleration)

| | |
|---|---|
| Description: | Sets deceleration of given axes.<br><br>The DEC setting only takes effect when the given axis is in closed-loop operation (servo mode ON).<br><br>DEC can be changed while the axis is moving. |
| Format: | DEC {<AxisID> <Deceleration>} |
| Arguments: | <AxisID> is one axis of the controller.<br><br><Deceleration> is the deceleration value in physical units/s$^2$. |
| Response: | None |
| Troubleshooting: | Illegal axis identifiers |
| Notes: | The lowest possible value for <Deceleration> is 0.<br><br>DEC changes the value of the ***Closed-Loop Deceleration (Phys. Unit/s$^2$)*** parameter (ID 0xC) in the volatile memory (can be saved as default with WPA (p. 245), can also be changed with SPA (p. 222) and SEP (p. 226)).<br><br>The maximum value that can be set with the DEC command is given by the ***Maximum Closed-Loop Deceleration (Phys. Unit/s$^2$)*** parameter (ID 0x4B) (can be changed with SPA (p. 226) and SEP (p. 222)). |

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

| | |
|---|---|
| Description: | Gets the deceleration value for closed-loop operation set with DEC.<br><br>If all arguments are omitted, gets the value of all axes set with DEC. |
| Format: | DEC? [{<AxisID>}] |

| | |
|---|---|
| Arguments: | <AxisID> is one axis of the controller. |
| Response: | {<AxisID>"="<float> LF}<br><br>where<br><br><float> is the deceleration value for closed-loop operation 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).<br>See the MAC command (p. 205) and "Controller Macros" (p. 121) for more information. |

**DFH (Define Home Position)**

| | |
|---|---|
| Description: | Redefines the zero position of the given axis by setting the position value to zero at the current position.<br><br>If all arguments are omitted, 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. 216) (query of the current position)
- TMN? (p. 237) (query of the minimum commandable position)
- TMX? (p. 237) (query of the maximum commandable position)

DFH does **not** change the values of the parameters for the definition of travel range and soft limits (p. 33).

The offset is reset to zero in the following cases:

- When switching on and rebooting the C-863:
  For all axes
- During reference point definition:
  For the axis in question

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. Then 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 the new zero position using DFH. As a result, the offset value for axis 1 is now 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: | Gets the sensor position at which the given axis has its zero position.<br><br>If all arguments are omitted, the position value of all axes is queried. |
| Format: | DFH? [{<AxisID>}] |
| Arguments: | <AxisID> is one axis of the controller. |
| Response: | {<AxisID>"="<SensorPosition> LF}<br><br>where<br><br><SensorPosition> is the sensor position which was valid at the time the last DFH command was processed. This sensor position value is used internally as offset for the calculation of the current axis position. |
| Troubleshooting: | Illegal axis identifier |
| Notes: | The sensor position which was valid when the last DFH command was processed is available in the volatile memory as an offset. The offset is reset to zero in the following cases: |

- When switching on and rebooting the C-863: for all axes
- During reference point definition: for the axis in question

See DFH for an example.

### DIO (Set Digital Output Line)

| | |
|---|---|
| Description: | Switches the specified digital output line(s) to specified state(s).<br><br>Use TIO? (p. 236) 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.<br><br><OutputOn> is the state of the digital output line, see below for details. |
| Response: | none |
| Notes: | Using the DIO command, you can activate/deactivate the Output 1 to Output 4 lines which are located on the I/O socket (p. 298). With the C-863, you can either set a single line per DIO command, or all lines at once.<br><br>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 given by <OutputOn>.<br><br>If <OutputOn>=1 the line is set to HIGH/ON, if <OutputOn>=0 it is set to LOW/OFF.<br>Do not use DIO on output lines for which the trigger output is enabled with TRO (p. 238). |

### DIO? (Get Digital Input Lines)

| | |
|---|---|
| Description: | Gets the states of the specified digital input lines.<br><br>Use TIO? (p. 236) 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}<br><br>where<br><br><InputOn> gives the state of the digital input line, see below for details. |
| Notes: | You can use the DIO? command to directly read the digital input lines 1 to 4 that are located on the **I/O** socket (p. 298).<br><br>The <DIOID> identifiers to use for the lines are 1 to 4. If the identifier is omitted or 0, all lines are queried.<br><br>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 kind of data to be recorded (record option) for the data recorder table given. |
| Format: | DRC {<RecTableID> <Source> <RecOption>} |

| | |
|---|---|
| Arguments: | <RecTableID>: is one data recorder table of the controller, see below.<br><br><Source>: is the data source, for example, an axis or a channel of the controller. The required source depends on the selected record option.<br><br><RecOption>: is the kind of data to be recorded (record option).<br><br>See below for a list of the available record options and the corresponding data sources. |
| Response: | None |
| Notes: | The C-863 has two data recorder tables with 1024 points per table.<br><br>With HDR? (p. 189) you will obtain a list of available record and trigger options and additional information about data recording. The number of available data recorder tables can be read with TNR? (p. 238).<br><br>For more information see "Data Recorder" (p. 95). |
| Recording options available with the corresponding data sources: | 0=Nothing is recorded<br>1=Commanded position of axis<br>2=Actual position of axis<br>3=Position error of axis<br>70=Commanded velocity of axis<br>71=Commanded acceleration of axis<br>73=Motor output of axis<br>74=Kp of axis<br>75=Ki of axis<br>76=Kd of axis<br>80=Signal status register of axis<br>81=Analog input (channel = 1 - 9)<br>90=Active parameter set of axis<br>91=Motor current of axis |

Note: The input channels for the record option 81 can be the Input 1 to 4 lines of the **I/O** socket (p. 298) (use IDs 1 to 4 for these data sources).

The data source IDs 5 to 7 refer to the inputs for the joystick axes and the joystick button:
5 = axis 1 of joystick device 1
6 = button 1 of joystick device1
7 = axis 2 of joystick device 2

Source IDs 7 and higher are reserved for additional analog input channels.

## DRC? (Get Data Recorder Configuration)

| | |
|---|---|
| Description: | Gets the settings made with DRC (p. 175). |
| Format: | DRC? [{<RecTableID>}] |
| Arguments: | <RecTableID>: is a data recorder table of the controller; if this information is omitted, the response will contain the settings for all tables. |
| Response: | The current DRC settings: |

{<RecTableID>"="<Source> <RecOption> LF}

where

<Source>: is the data source, for example an axis or a channel of the controller. The source type depends on the record option.

<RecOption>: is the kind of data to be recorded (record option).

See DRC for a list of the available record options and the corresponding data sources.

### DRR? (Get Recorded Data Values)

| | |
|---|---|
| Description: | Gets the last recorded data.<br><br>Reading can take some time depending on the number of points to be read!<br><br>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.<br><br><NumberOfPoints> is the number of points to be read per table.<br><br><RecTableID> is one data recorder table of the controller. |
| Response: | For the recorded data in GCS array format, see the separate manual for the GCS array, SM146E, and the example below. |
| Notes: | If <RecTableID> is omitted, the data from all tables with non-zero record option is read.<br><br>With HDR? (p. 189) you will obtain a list of available record options and trigger options and additional information about data recording.<br><br>For further information see the description of the DRC command (p. 175) and "Data Recorder" (p. 95). |

Example:

```
rtr?
10
drr? 1 20
# REM C-863
#
# 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
4.99998 0.00002
5.00000 0.00000
4.99998 0.00002
5.00000 0.00000
5.00000 0.00000
5.00000 0.00000
5.00000 0.00000
4.99998 0.00002
5.00000 0.00000
4.99998 0.00002
4.99998 0.00002
5.00000 0.00002
4.99998 0.00004
```

## DRT (Set Data Recorder Trigger Source)

| | |
|---|---|
| Description: | Defines a trigger source for the given data recorder table. |
| Format: | DRT <RecTableID> <TriggerSource> <Value> |
| Arguments: | <RecTableID> is one data recorder table of the controller. See below for details.<br><br><TriggerSource> ID of the trigger source, see below for a list of available options.<br><br><Value> depends on the trigger source, can be a dummy, see below. |
| Response: | none |
| Notes: | At present, only 0 is valid for <RecTableID>; this means that the given trigger source is set for all data recorder tables.<br><br>Regardless of the trigger option set, the data recording is always triggered when a step response measurement is started with STE (p. 231).<br><br>A trigger option set with DRT will become valid for all data recorder tables with non-zero record option.<br><br>With HDR? (p. 189) you will obtain a list of available record options and trigger options and additional information about data recording.<br><br>For further information see the description of the DRC command (p. 175) and "Data Recorder" (p. 95). |
| Available trigger options: | 0 = default setting; data recording is triggered with STE; <Value> must be a dummy<br><br>1 = any command changing target position (e.g. MVR (p. 213), MOV (p. 211)); <Value> must be a dummy<br><br>2 = next command, resets trigger after execution; <Value> must be a dummy |

|  |  |
|---|---|
|  | 6 = any command changing target position (e.g. MVR, MOV), resets trigger after execution; <Value> must be a dummy |
|  | 7 = SMO command, 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}<br><br>where<br><br><TriggerSource> is the identifier of the trigger source.<br><br><Value> depends on the trigger source; 0 is a dummy.<br><br>Further information is found in the description of the DRT command (p. 180). |
| Notes: | Since all data recorder tables of the C-863 have the same trigger source, the DRT? response is given as a single line of the form<br><br>0=<TriggerSource> <Value> |

### ERR? (Get Error Number)

| | |
|---|---|
| Description: | Gets error code <int> of the last occurred error and resets the error to 0.<br><br>Only the last error is buffered. You should therefore call ERR? after each command.<br><br>The error codes and their descriptions are listed in "Error Codes" (p. 247). |
| Format: | ERR? |
| Arguments: | None |
| Response: | The error code of the last error that occurred (integer). |
| Troubleshooting: | Communication breakdown |

### FED (Find Edge)

| | |
|---|---|
| Description: | Moves given axis to a given signal edge.<br><br>FED does not set a certain position value at the selected edge (in contrast to the FNL (p. 184), FPL (p. 185) and FRF (p. 187) commands for reference point definition), i.e. the axis is not "referenced" after using FED.<br>If multiple axes are given in the command, they are moved synchronously. |
| Format: | FED {<AxisID> <EdgeID> <Param>} |
| Arguments: | <AxisID> is one axis of the controller.<br><br><EdgeID> is the type of edge the axis has to move to. See below for available edge types.<br><br><Param> depends on the selected edge and qualifies it. See below for details. |
| Response: | none |

| | |
|---|---|
| Notes: | Servo mode must be switched on with SVO (p. 233) for the commanded axis prior to using this command (closed-loop operation).<br>The C-863 firmware detects the presence or absence of reference point switch and limit switches using parameters (ID 0x14 for reference point switch; ID 0x32 for limit switches). According to the values of those parameters, the C-863 enables or disables FED motions to the appropriate signal edges. Adapt the parameter values to your hardware using SPA (p. 226) or SEP (p. 222). See "Parameter Overview" (p. 272) for more information.<br>You can use the digital input lines instead of the switches as source of the switch signals for FED. For further information see "Digital Input Signals" (p. 106).<br>FED can be used to measure the physical travel range of a new mechanics and thus to identify the values for the corresponding parameters: the distance from negative to positive limit switch, the distance between the negative limit switch and the reference point switch (parameter ID 0x17), and the distance between reference point switch and positive limit switch (parameter ID 0x2F). For further information see "Travel Range and Soft Limits" (p. 33).<br><br>The motion can be stopped by #24 (p. 156), STP (p. 232) and HLT (p. 191).<br><br>Motion commands like FED are not allowed when the joystick is active for the axis. For further information see "Joystick Control" (p. 112). |
| Available edge types and parameters: | The following edge types with their parameter settings are available:<br><br>1 = negative limit switch, <Param> must be 0<br>2 = positive limit switch, <Param> must be 0<br>3 = reference point switch, <Param> must be 0 |

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

| | |
|---|---|
| Description: | Starts a reference move.<br><br>Moves the given axis to the negative physical limit of its travel range and sets the current position to a defined value. See below for details.<br><br>If multiple axes are given in the command, they are moved synchronously. |
| Format: | FNL [{<AxisID>}] |
| Arguments: | <AxisID> is one axis of the controller, if omitted, all axes are involved. |
| Response: | none |
| Troubleshooting: | Illegal axis identifier |
| Notes: | Servo mode must be switched on with SVO (p. 233) for the commanded axis prior to using this command (closed-loop operation).<br>If reference move was successful, absolute motion will afterwards be possible in closed-loop operation.<br>With the C-863, the negative limit switch of the mechanical system is used to determine the negative physical limit of the travel range. 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 (value can be negative).<br><br>You can use a digital input instead of the negative limit switch as source of the negative limit switch signal for FNL. Further information see "Digital Input Signals" (p. 106).<br><br>The motion can be stopped by #24 (p. 156), STP (p. 232) and HLT (p. 191).<br><br>Use FRF? (p. 188) to check whether the reference move was successful.<br><br>Use FRF (p. 187) instead of FNL to perform a reference |

move for an axis that has no limit switches but a reference point switch.

For best repeatability, always perform the reference point definition 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. The FNL and FPL commands will provoke an error message, and only the reference point switch can be used for a reference move (FRF).

Further information, see "Travel Range and Soft Limits" (p. 33).

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

| | |
|---|---|
| Description: | Starts a reference move.<br><br>Moves the given axis to the positive physical limit of its travel range and sets the current position to a defined value. See below for details.<br><br>If multiple axes are given in the command, they are moved synchronously. |
| Format: | FPL [{<AxisID>}] |
| Arguments: | <AxisID> is one axis of the controller, if omitted, all axes are involved. |
| Response: | none |
| Troubleshooting: | Illegal axis identifier |
| Notes: | Servo mode must be switched on with SVO (p. 233) for the commanded axis prior to using this command (closed-loop operation).<br>If reference move was successful, absolute motion will afterwards be possible in closed-loop operation. |

With the C-863, the positive limit switch of the mechanical systems is used to determine the positive physical limit of the travel range. 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. Further information see "Digital Input Signals" (p. 106).

The motion can be stopped by #24 (p. 156), STP (p. 232) and HLT (p. 191).

Use FRF? (p. 188) to check whether the reference move was successful.

Use FRF (p. 187) instead of FPL to perform a reference move for an axis that has no limit switches but a reference point switch.

For best repeatability, always perform the reference point definition 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. The FNL and FPL commands will provoke an error message, and only the reference point switch can be used for a reference move (FRF).

Further information, see "Travel Range and Soft Limits" (p. 33).

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

| | |
|---|---|
| Description: | Starts a reference move.<br><br>Moves the given axis to the reference point switch and sets the current position to a defined value. See below for details.<br><br>If multiple axes are given in the command, they are moved synchronously. |
| Format: | FRF [{<AxisID>}] |
| Arguments: | <AxisID> is one axis of the controller, if omitted, all axes are involved. |
| Response: | none |
| Troubleshooting: | Illegal axis identifier |
| Notes: | Servo mode must be switched on with SVO (p. 233) for the commanded axis prior to using this command (closed-loop operation).<br>If reference move was successful, absolute motion will afterwards be possible in closed-loop operation.<br>The value of the parameter 0x16 is set as the current position when the axis is at the reference point switch.<br><br>You can use a digital input instead of the reference point switch as source of the reference signal for the FRF command. Further information see "Digital Input Signals" (p. 106).<br><br>The motion can be stopped by #24 (p. 156), STP (p. 232) and HLT (p. 191).<br><br>Use FRF? (p. 188) to check whether the reference move was successful.<br><br>Use FNL (p. 184) or FPL (p. 185) instead of FRF (p. 187) to perform a reference move for an axis that has no reference point switch but limit switches. |

For best repeatability, always perform the reference point definition in the same way. The FRF command always approaches the reference point switch from the same side, no matter where the axis is when the command is called.

Further information, see "Travel Range and Soft Limits" (p. 33).

### FRF? (Get Referencing Result)

| | |
|---|---|
| Description: | Gets whether the given axis is referenced or not. |
| Format: | FRF? [{<AxisID>}] |
| Arguments: | <AxisID> is one axis of the controller. |
| Response: | {<AxisID>"="<uint> LF}<br><br>where<br><br><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 performed with FNL (p. 184), FPL (p. 185) or FRF (p. 187) or when the position was set directly with POS (p. 215) (depending on the mode of reference point definition set with RON (p. 217)). |

### GOH (Go To Home Position)

| | |
|---|---|
| Description: | Moves the given axes to the zero position.<br><br>GOH [{<AxisID>}]<br>is identical to<br>MOV {<AxisID> 0}<br><br>Servo mode must be switched on for the commanded axis prior to using this command (closed-loop operation).<br><br>The motion can be stopped by #24 (p. 156), STP (p. 232) and HLT (p. 191). |
| Format: | GOH [{<AxisID>}] |
| Arguments: | <AxisID>: is an axis of the controller, if this is omitted, 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 about data recording (record options and trigger options, information about additional parameters and commands concerned with data recording). |
| Format: | HDR? |
| Arguments: | None |
| Response | #RecordOptions<br>{<RecOption>"="<DescriptionString>[ of <Channel>]}<br><br>#TriggerOptions<br>[{<TriggerOption>"="<DescriptionString>}]<br><br>#Parameters to be set with SPA<br>[{<ParameterID>"="<DescriptionString>}] |

#Additional information
[{<Command description>"("<Command>")"}]

#Sources for Record Options
[{<RecOption>"="<Source>}]

end of help

Example: For the C-863, 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)
90=active parameterset
91=Motorcurrent
#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
7=with SMO command, reset trigger after
execution
#Additional information
2 record tables
1024 datapoints per table
end of help
```

| | |
|---|---|
| | Note: TriggerOptions = 0 (default) means that recording is triggered by the STE command (p. 231). |

### 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: | Halts the motion of given axes smoothly. For details see the notes below.<br><br>Error code 10 is set.<br><br>#24 (p. 156) and STP (p. 232) 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 omitted all axes are halted |
| Response: | none |
| Troubleshooting: | Illegal axis identifier |
| Notes: | HLT stops motion with given system deceleration with regard to system inertia.<br><br>HLT stops all motion caused by motion commands (e.g. MOV (p. 211), MVR (p. 213), GOH (p. 189), STE (p. 231), SMO (p. 224)), commands for reference point definition (FNL (p. 184), FPL (p. 185), FRF (p. 187)) and macros |

(MAC (p. 205)).

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 which contains all available parameters with short descriptions. For further information, see "Parameter Overview" (p. 272). |
| Format: | HPA? |
| Arguments: | None |
| Response | {<PamID>"="<string> LF} |

where

<PamID> is the ID of one parameter, hexadecimal format

<string> is a string which describes the corresponding parameter.

The string has following format:

<CmdLevel>TAB<MaxItem>TAB<DataType>TAB<Function GroupDescription>TAB<ParameterDescription>[{TAB<PossibleValue>"="<ValueDescription>}]

where

<CmdLevel> is the command level which 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-863, an "item" is an axis.

<DataType> is the data type of the parameter value, can be INT, FLOAT or CHAR

<FunctionGroupDescription> is the name of the function group to which the parameter belongs

<ParameterDescription> is the parameter name

<PossibleValue> is one value from the allowed data range

<ValueDescription> is the meaning of the corresponding value

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

SPA (p. 226) influences the parameter settings in volatile memory (RAM).

WPA (p. 245) copies parameter settings from volatile to nonvolatile memory.

SEP (p. 222) writes parameter settings directly into nonvolatile memory (without changing settings in volatile memory).

RPA (p. 218) resets volatile memory to the values from nonvolatile memory.

### JAS? (Query Joystick Axis Status)

| | |
|---|---|
| Description: | Gets the current status of the given axis of the given joystick device which is connected to the controller. |
| Format: | JAS? [{<JoystickID> <JoystickAxis>}] |

|  |  |
|---|---|
| Arguments: | <JoystickID> is one joystick device connected to the controller; see below for details.<br><br><JoystickAxis> is one of the axes of the joystick device; see below for details. |
| Response: | {<JoystickID> <JoystickAxis>"="<Amplitude>}<br><br>where<br><br><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: | Two joystick devices can be connected to the **Joystick** socket of the C-863, the identifiers are 1 and 2. The C-863 supports one axis per joystick device, the identifier of the joystick axis is 1 in each case. For further information see "Commandable Items" (p. 18).<br><br>The <Amplitude> factor is applied to the velocity set with VEL (p. 242), the range is -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. 197) for an example). |

### JAX (Set Axis Controlled By Joystick)

|  |  |
|---|---|
| Description: | Sets axis controlled by a joystick which is connected to the controller.<br><br>Each axis of the controller can only be controlled by one joystick axis. |
| Format: | JAX <JoystickID> <JoystickAxis> <AxisID> |

| | |
|---|---|
| Arguments: | <JoystickID> is one joystick device connected to the controller; see below for details.<br><br><JoystickAxis> is one of the axes of the joystick device; see below for details.<br><br><AxisID> is one axis of the controller. |
| Response: | none |
| Notes: | Two joystick devices can be connected to the **Joystick** socket of the C-863, the identifiers are 1 and 2. The C-863 supports one axis per joystick device, the identifier of the joystick axis is 1 in each case. For further information see "Commandable Items" (p. 18). |

### JAX? (Get Axis Controlled By Joystick)

| | |
|---|---|
| Description: | Gets axis controlled by a joystick which is connected to the controller. |
| Format: | JAX? [{<JoystickID> <JoystickAxis>}] |
| Arguments: | <JoystickID> is one joystick device connected to the controller; see below for details.<br><br><JoystickAxis> is one of the axes of the joystick device; see below for details. |
| Response: | {<JoystickID> <JoystickAxis>"="{<AxisID> }LF}<br><br>where<br><br><AxisID> is one axis of the controller. |
| Notes: | Two joystick devices can be connected to the **Joystick** socket of the C-863, the identifiers are 1 and 2. The C-863 supports one axis per joystick device, the identifier of the joystick axis is 1 in each case. For further information see "Commandable Items" (p. 18). |

### JBS? (Query Joystick Button Status)

| | |
|---|---|
| Description: | Gets the current status of the given button of the given joystick device which is connected to the controller. |
| Format: | JBS? [{<JoystickID> <JoystickButton>}] |
| Arguments: | <JoystickID> is one joystick device connected to the controller; see below for details.<br><br><JoystickButton> is one of the buttons of the joystick device; see below for details. |
| Response: | {<JoystickID> <JoystickButton> "="<State>}<br><br>where<br><br><State> indicates if the joystick button is pressed; 0 = not pressed, 1 = pressed. |
| Notes: | Two joystick devices can be connected to the **Joystick** (p. 299) socket of the C-863, the identifiers are 1 and 2. The C-863 supports one button of joystick device 1, the identifier of the joystick button is 1. See also "Commandable Items" (p. 18). |

### JDT (Set Joystick Default Lookup Table)

| | |
|---|---|
| Description: | Sets lookup table type for the given axis of the given joystick device which is connected to the controller.<br><br>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 device connected to the controller; see below for details.<br><br><JoystickAxis> is one of the axes of the joystick device; see below for details.<br><br><uint> defines the type of lookup-table profile to use; see below for details. |
| Response: | none |
| Notes: | Two joystick devices can be connected to the **Joystick** socket of the C-863, the identifiers are 1 and 2. The C-863 supports one axis per joystick device, the identifier of the joystick axis is 1 in each case. For further information see "Commandable Items" (p. 18).<br>Note that the number of write cycles in the non-volatile memory is limited. Change the lookup table type only if necessary. |
| Available lookup tables: | The C-863 provides the following types for the lookup-table profile:<br><br>1 = linear (default)<br>2 = parabolic |

### JLT (Fill Joystick Lookup Table)

| | |
|---|---|
| Description: | Fills the lookup table for the given axis of the given joystick device which is connected to the controller.<br><br>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. 196).<br><br>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 device connected to the controller; see below for details.<br><br><JoystickAxis> is one of the axes of the joystick device; see below for details.<br><br><Addr> is the index of a point in the lookup table, starts with 1.<br><br><floatn> is the value of point n. |
| Response: | none |
| Notes: | Two joystick devices can be connected to the **Joystick** socket of the C-863, the identifiers are 1 and 2. The C-863 supports one axis per joystick device, the identifier of the joystick axis is 1 in each case. See also "Commandable Items" (p. 18).<br><br>The <floatn> values are factors which will during joystick control be applied to the velocity set with VEL (p. 242), the range is -1.0000 to 1.0000.<br><br>The <floatn> values are automatically written to the non-volatile memory of the C-863.<br><br>Example: In the current lookup table, point 1 has the value -1, hence the controlled axis will move with full velocity in negative direction at the maximum negative displacement of the joystick. Points 124 to 133 have the value 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 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 that the number of write cycles in the non-volatile memory is limited. Write values to the lookup table only if necessary.

For further information see "Joystick Control" p. 112.

### JLT? (Get Joystick Lookup Table Values)

| | |
|---|---|
| Description: | Gets 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<br><br><NumberOfPoints>: is the number of points to be read per joystick axis; maximum number is 256.<br><br><JoystickID> is one joystick device connected to the controller; see below for details.<br><br><JoystickAxis> is one of the axes of the joystick device; 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-863, <JoystickID> and <JoystickAxis> must be omitted in the JLT? command, while <StartPoint> and <NumberOfPoints> are always required.<br><br>The <floatn> values in the lookup table are factors which will during joystick control be applied to the velocity set with VEL (p. 242), 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: | Enables or disables a joystick device which is connected to the controller. |
| Format: | JON {<JoystickID> <uint>} |
| Arguments: | <JoystickID> is one joystick device connected to the controller; see below for details.<br><br><uint> 1 enables the joystick device, 0 disables joystick device. |
| Response: | none |
| Notes: | Two joystick devices can be connected to the **Joystick** socket (p. 299) of the C-863, the identifiers are 1 and 2. For further information see "Connecting an Analog Joystick" (p. 63).<br><br>Before a joystick device can be activated with JON, its axes must have been assigned to the controller axes using JAX (p. 194).<br><br>While a joystick connected to the C-863 is enabled with the JON command, this joystick controls the axis velocity ("commanded velocity" output by the profile generator). During joystick control, the target position is set to the soft limit given by parameter 0x15 or 0x30. For parameter details see "Travel Range and Soft Limits" (p. 33). In open-loop operation (servo mode switched off) no joystick control is possible. When disabling the joystick, the target position is set to the current position of the joystick-controlled axis.<br><br>Motion commands like MOV (p. 211) are not allowed when a joystick is active on the axis. For further information see "Joystick Control" (p. 112). |

### JON? (Get Joystick Activation Status)

| | |
|---|---|
| Description: | Gets activation state of the given joystick device which is connected to the controller. |
| Format: | JON? [{<JoystickID>}] |
| Arguments: | <JoystickID> is one joystick device connected to the controller; see below for details. |
| Response: | {<JoystickID>"="<uint>}<br><br>where<br><br><uint> is the joystick activation state: 1 = joystick device enabled, 0 = joystick device disabled. |
| Notes: | Two joystick devices can be connected to the **Joystick** socket of the C-863, the identifiers are 1 and 2. For further information see "Connecting an Analog Joystick" (p. 63). |

### JRC (Jump Relatively Depending On Condition)

| | |
|---|---|
| Description: | Jumps relatively depending on a given condition of the following type: one given value is compared with a queried value according to a given rule.<br><br>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. 244). 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 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 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 (ONT? query). When the stop button is pressed as long as the target position has not been reached yet: The response to the POS? 1 query is copied to the TARGET variable. Then this variable is used as second argument for the MOV command. Thus the stage stays where it just 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: | Gets whether axes have limit switches. |
| Format: | LIM? [{<AxisID>}] |
| Arguments: | <AxisID>: is one axis of the controller |
| Response: | {<AxisID>"="<uint> LF}<br><br>where<br><br><uint> indicates whether the axis has limit switches (=1) or not (=0). |
| Troubleshooting: | Illegal axis identifier |
| Notes: | The C-863 firmware detects the presence or absence of limit switches using a parameter (ID 0x32). According to the value of this parameter, the C-863 enables or disables the stopping of the motion at the limit switches and reference moves using FNL (p. 184) or FPL (p. 185).<br><br>Adapt the parameter value to your hardware using SPA (p. 226) or SEP (p. 222). For further information see "Limit Switch Detection" (p. 32).<br><br>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. 106). |

### MAC (Call Macro Function)

| | |
|---|---|
| Description: | Calls a macro function. Permits recording, deleting and running macros on the controller. |
| Format: | MAC <keyword> {<parameter>}<br><br>in particular:<br><br>MAC BEG <macroname><br>MAC DEF <macroname><br>MAC DEF?<br>MAC DEL <macroname><br>MAC END<br>MAC ERR?<br>MAC NSTART <macroname> <uint> [<String1> [<String2>]]<br>MAC START <macroname> [<String1> [<String2>]] |
| Arguments: | <keyword> determines which macro function is called. The following keywords and parameters are used:<br><br>MAC BEG <macroname><br>Starts recording a macro to be named *macroname* on the controller; may not be used in a macro; the commands that follow become the macro. End the recording with MAC END. Note that erroneous macro content cannot be detected by sending the ERR? command.<br><br>MAC END<br>Stops macro recording (cannot become part of a macro) |

MAC ERR?

Reports the last error which occurred during macro execution.

Response: <macroname> <uint1>”=”<uint2> <”<”CMD”>”>

where <macroname> is the name of the macro, <uint1> is the line in the macro, <uint2> is the error code and <”<”CMD”>”> is the erroneous command which was sent to the parser.

MAC DEF <macroname>

Sets specified macro as start-up macro. This macro will be automatically executed with the next switching-on or reboot of the controller. If <macroname> is omitted, the current start-up macro selection is canceled.

MAC DEF?

Asks for the start-up macro
Response: <macroname>
If no start-up macro is defined, the response is an empty string with the terminating character.

MAC DEL <macroname>

Deletes specified macro.

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

Repeats the specified macro <uint> times. Another execution is started when the last one is finished.

<String1> and <String2> are optional arguments which give the values for local variables 1 and 2 used in the given macro. <String1> and <String2> can be given directly or via the values of variables. Macro execution will fail if the macro contains local variables but <String1> and <String2> are omitted in the MAC NSTART command. See “Variables” (p. 148) for further details.

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

Starts one execution of the specified macro. <String1> and <String2> have the same function as with MAC NSTART.

| | |
|---|---|
| Response: | None |
| Troubleshooting: | Macro recording is active (keywords BEG, DEL) or inactive (END)<br>Macro contains a disallowed MAC command |
| Notes: | During macro recording no macro execution is allowed.<br><br>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. Further information see "Working with Macros" (p. 123) and "Target and Sender Address" (p. 146).<br><br>When macros are recorded on the ***Controller macros*** tab in PIMikroMove, the `MAC BEG` and `MAC END` commands must be omitted.<br><br>A macro can be overwritten by a macro with the same name.<br><br>Macros can contain local and global variables. For more information, see "Variables" (p. 148).<br><br>A running macro sends no responses to any interface.<br><br>Depending on the value of parameter 0x72 (***Ignore Macro Error?***), the following options exist when an error is caused by a running macro:<br><br>0 = Macro execution is aborted (default).<br>1 = The error is ignored and the macro execution is continued.<br><br>Irrespective of the parameter setting, MAC ERR? always reports the last error that occurred during a macro execution. |

The following commands provided by the C-863 can only be used in macros:

DEL (p. 171), JRC (p. 202), MEX (p. 209) and WAC (p. 244).

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 when 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. 156) and STP (p. 232).

Simultaneous execution of multiple macros is not possible. Only one macro can be executed at a time.

A running macro may not be deleted.

You can query with #8 (p. 156) 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 given macro. |
| Format: | MAC? [<macroname>] |
| Arguments | <macroname>: name of the macro whose content shall be listed; if omitted, the names of all stored macros are listed. |

| | |
|---|---|
| Response: | <string><br><br>If <macroname> was given, <string> is the content of this macro;<br><br>If <macroname> was omitted, <string> is a list with the names of all stored macros |
| Troubleshooting: | Macro <macroname> not found |

## MEX (Stop Macro Execution Due To Condition)

| | |
|---|---|
| Description: | Stops macro execution due to a given condition of the following type: a given value is compared with a queried value according to a given rule.<br><br>Can only be used in macros.<br><br>When the macro interpreter accesses this command, the condition is checked. If it is true, the current macro is stopped; otherwise macro execution is continued with the next line. Should the condition be fulfilled later, the interpreter will ignore it.<br><br>See also the WAC command (p. 244). |
| 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.<br><br><OP> is the operator to be used. The following operators are possible:<br>= <= < > >= !=<br>Important: There must be a blank space before and after the operator!<br><br><Value> is the value that is compared with the response to <CMD?>. |

| | |
|---|---|
| Response: | None |
| Example: | Send: `MAC START LOOP` |

Note: Macro LOOP has the following contents:

```
MAC START KEY1
MAC START KEY2
MEX DIO? 4 = 1
MAC START LOOP
```

Macro KEY1 has the following contents:

```
MEX DIO? 4 = 1
MEX DIO? 1 = 0
MVR 1 1.0
DEL 100
```

Macro KEY2 has the following content:

```
MEX DIO? 4 = 1
MEX DIO? 2 = 0
MVR 1 -1.0
DEL 100
```

Macro LOOP forms an infinite loop by permanently calling KEY1, KEY2 and itself.

KEY1 checks the state of the digital input channel 1 (located on the **I/O** socket (p. 298)). 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.

Connecting the digital input channels 1, 2 and 4 with pushbuttons, e.g. with the C-170.PB pushbutton box, it is possible to implement interactive control of an axis without any software assistance. The delay (DEL 100) is required to avoid generation of multiple MVR commands while pressing the pushbutton for a short time.

Channel 4 is used as a global exit. Since MEX stops execution of the current macro only, it must also be included in the calling macro, which would otherwise continue.

### MOV (Set Target Position)

| | |
|---|---|
| Description: | Set new absolute target position for given axis.<br><br>Servo mode must be switched on for the commanded axis prior to using this command (closed-loop operation). |
| Format: | MOV {<AxisID> <Position>} |
| Arguments | <AxisID> is one axis of the controller<br><br><Position> is the new absolute target position in physical units. |
| Response: | none |
| Notes: | The target position must be inside the soft limits. Use TMN? (p. 237) and TMX? (p. 237) to ask for the current valid soft limits.<br><br>The motion can be stopped by #24 (p. 156), STP (p. 232) and HLT (p. 191).<br><br>During a motion, a new motion command resets the target to a new value and the old one 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 may overwrite each other. |

Motion commands like MOV are not allowed when a joystick is active on the axis. For further information see "Joystick Control" (p. 112).

| | |
|---|---|
| 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. 182) indicates that the target position given in the motion command is out of limits. |

### MOV? (Get Target Position)

| | |
|---|---|
| Description: | Returns last valid commanded target position. |
| Format: | MOV? [{<AxisID>}] |
| Arguments: | <AxisID> is one axis of the controller |
| Response: | {<AxisID>"="<float> LF}<br><br>where<br><br><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. 211), MVR (p. 213), GOH (p. 189), STE (p. 231)) or by the joystick (when disabling a joystick, the target position is set to the current position for joystick-controlled axes in closed-loop operation).<br><br>MOV? gets the commanded positions. Use POS? (p. 216) to get the current positions. |

### MVR (Set Target Relative To Current Position)

| | |
|---|---|
| Description: | Moves the given axis relative to the last commanded target position.<br><br>Servo mode must be switched on for the commanded axis prior to using this command (closed-loop operation). |
| Format: | MVR {<AxisID> <Distance>} |
| Arguments: | <AxisID> is one axis of the controller.<br><br><Distance> gives the distance that the axis is to move; the sum of the distance and the last commanded target position is set as new target position (in physical units). |
| Response: | None |
| Notes: | The target position must be inside the soft limits. Use TMN? (p. 237) and TMX? (p. 237) to ask for the current soft limits, and MOV? (p. 212) for the current target.<br><br>The motion can be stopped by #24 (p. 156), STP (p. 232) and HLT (p. 191).<br><br>During a motion, a new motion command resets the target to a new value and the old one 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 may overwrite each other.<br><br>Motion commands like MVR are not allowed when a joystick is active on the axis. For further information see "Joystick Control" (p. 112). |

| | |
|---|---|
| Example: | Send: `MOV 1 0.5`<br>Note: This is an absolute motion.<br>Send: `POS? 1`<br>Receive: `1=0.500000`<br>Send: `MOV? 1`<br>Receive: `1=0.500000`<br>Send: `MVR 1 2`<br>Note: This is a relative motion.<br>Send: `POS? 1`<br>Receive: `1=2.500000`<br>Send: `MVR 1 2000`<br>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.<br>Send: `MOV? 1`<br>Receive: `1=2.500000`<br>Send: `POS? 1`<br>Receive: `1=2.500000` |

**ONT? (Get On-Target State)**

| | |
|---|---|
| Description: | Gets on-target state of given axis.<br><br>If all arguments are omitted, gets state of all axes. |
| Format: | ONT? [{<AxisID>}] |
| Arguments: | <AxisID> is one axis of the controller. |
| Response: | {<AxisID>"="<uint> LF}<br><br>where<br><br><uint> = "1" when the specified axis is on target, "0" otherwise. |
| Troubleshooting: | Illegal axis identifier |

| | |
|---|---|
| Notes: | The detection of the on-target state is only possible in closed-loop operation (servo mode ON).<br><br>The on-target state is influenced by the settings for the settling window (parameter 0x36) and the delay time (parameter 0x3F). Details see "On-Target State" (p. 30). |

### POS (Set Real Position)

| | |
|---|---|
| Description: | Sets the current position (does not cause motion). |
| Format: | POS { <AxisID> <Position>} |
| Arguments: | <AxisID> is one axis of the controller.<br><br><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 mode of reference point definition is set to "0", see RON (p. 217).<br><br>An axis is considered to be "referenced" when the position has been set with POS (for more information, see "Reference Point Definition" (p. 38)).<br><br>The minimum and maximum commandable positions (TMN? (p. 237), TMX? (p. 237)) are not adapted when a position is set with POS. This can result in target positions which are allowed by the C-863 but cannot be reached by the hardware. Target positions are also possible that can be reached by the hardware but are refused by the C-863. Furthermore, the zero position can be outside of the physical travel range after using POS. |

### POS? (Get Real Position)

|  |  |
|---|---|
| Description: | Gets the current axis position.<br><br>If all arguments are omitted, the current position of all axes is queried. |
| Format: | POS? [{<AxisID>}] |
| Arguments: | <AxisID> is one axis of the controller. |
| Response: | {<AxisID>"="<float> LF}<br><br>where<br><br><float> is the current axis position in physical units. |
| Troubleshooting: | Illegal axis identifier |

### RBT (Reboot System)

|  |  |
|---|---|
| Description: | Reboots system. Controller behaves just like after switching-on. |
| Format: | RBT |
| Arguments: | none |
| Response: | none |
| Notes: | RBT cannot be used in macros. This is to avoid problems with start-up macro execution. |

### RMC? (List Running Macros)

|  |  |
|---|---|
| Description: | Lists macros which are currently running. |
| Format: | RMC? |
| Arguments: | none |

| | |
|---|---|
| Response: | {<macroname> LF}<br><br>where<br><br><macroname> is the name of one macro which is saved on the controller and currently running. The response is an empty line when no macro is running. |

## RON (Set Reference Mode)

| | |
|---|---|
| Description: | Sets mode of reference point definition of given axes. |
| Format: | RON {<AxisID> <ReferenceOn>} |
| Arguments: | <AxisID> is one axis of the controller.<br><br><ReferenceOn> can be 0 or 1. 1 is default. Details see below. |
| Response: | None |
| Troubleshooting: | Illegal axis identifier |
| Notes: | <ReferenceOn> = 0: To define the reference point of the axis, an absolute position value can be assigned with POS (p. 215) or a reference move can be started with FRF (p. 187), FNL (p. 184) or FPL (p. 185). Relative motions with MVR are possible, even when the reference point for the axis has not been defined yet.<br><br><ReferenceOn> = 1: To define the reference point of the axis, a reference move must be started with FRF, FNL or FPL. Using POS is not permitted. Motions in closed-loop operation are only possible when the reference point for the axis has been defined.<br><br>Further information, see "Reference Point Definition" (p. 38) and "Travel Range and Soft Limits" (p. 33). |

## RON? (Get Reference Mode)

| | |
|---|---|
| Description: | Gets mode of reference point definition of given axes. |
| Format: | RON? [{ <AxisID>}] |
| Arguments: | <AxisID> is one axis of the controller. |
| Response: | {<AxisID>"="<ReferenceOn> LF}<br><br>where<br><br><ReferenceOn> is the currently set mode of reference point definition for the axis |
| Troubleshooting: | Illegal axis identifier |
| Note: | You can find further information in the description of the RON command (p. 217). |

## RPA (Reset Volatile Memory Parameters)

| | |
|---|---|
| Description: | Resets the given parameter of the given item. The value from nonvolatile memory is written into volatile memory.<br><br>Related commands:<br><br>With HPA? (p. 192) you can obtain a list of the available parameters. SPA (p. 226) influences the parameter settings in volatile memory, WPA (p. 245) writes parameter settings from volatile to nonvolatile memory, and SEP (p. 222) writes parameter settings directly into nonvolatile memory (without changing the settings in volatile memory).<br><br>See SPA for an example. |
| Format: | RPA [{<ItemID> <PamID>}] |

| | |
|---|---|
| Arguments: | <ItemID> is the item for which a parameter is to be reset. See below for details.<br><br><PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details. |
| Response: | none |
| Troubleshooting: | Illegal item identifier, wrong parameter ID |
| Notes: | With the C-863, you can reset either all parameters or one single parameter with RPA. |
| Available item IDs and parameter IDs: | An item is an axis, the identifier can be changed with SAI (p. 220). For further information see "Commandable Items" (p. 18).<br><br>Valid parameter IDs are given in "Parameter Overview" (p. 272). |

### 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-863 is 50 µs |
| | the number of points for the C-863 is 1024 (length of data recorder table) |
| | For more information see "Data Recorder" (p. 95). |
| | 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 given 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<br><br><NewIdentifier> is the new identifier to use for the axis, see below for details |
| Response: | none |
| Notes: | An axis could be identified with up to 8 characters. Use TVI? (p. 240) to ask for valid characters.<br>The new axis identifier is saved automatically and thus still available after reboot or next switching-on. |

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

| | |
|---|---|
| Description: | Gets the axis identifiers.<br><br>See also "Commandable Items" (p. 18). |
| Format: | SAI? [ALL] |
| Arguments: | [ALL] is optional. For controllers which allow for axis deactivation, [ALL] ensures that the answer also includes the axes which are "deactivated". |
| Response: | {<AxisID> LF}<br><br><AxisID> is one axis of the controller. |

**SEP (Set Nonvolatile Memory Parameters)**

| | |
|---|---|
| Description: | Sets a parameter of a given item to a different value in nonvolatile memory, where it becomes the new default.<br><br>After parameters were set with SEP, you can use RPA (p. 218) to activate them (write them to volatile memory) without controller reboot.<br><br>**Note that this command is for setting hardware-specific parameters. Wrong values may lead to improper operation or damage of your hardware!**<br><br>Related commands:<br><br>HPA? (p. 192) returns a list of the available parameters.<br><br>SPA (p. 226) writes parameter settings into volatile memory (without changing the settings in nonvolatile memory).<br><br>WPA (p. 245) writes parameter settings from volatile to nonvolatile memory.<br><br>See SPA for an example. |
| Format: | SEP <Pswd> {<ItemID> <PamID> <PamValue>} |
| Arguments | <Pswd> is the password for writing to nonvolatile memory, default is "100".<br><br><ItemID> is the item for which a parameter is to be changed in 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.<br><br><PamValue> is the value to which the given parameter of the given item is set. |
| Response: | none |
| Troubleshooting: | Illegal item 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.**<br><br>With the C-863, you can write only one single parameter per SEP command. |
| Available item IDs and parameter IDs: | An item is an axis, the identifier can be changed with SAI (p. 220). For further information see "Commandable Items" (p. 18).<br><br>Valid parameter IDs are given in "Parameter Overview" (p. 272). |

**SEP? (Get Nonvolatile Memory Parameters)**

| | |
|---|---|
| Description: | Gets the value of a parameter of a given item from nonvolatile memory.<br><br>With HPA? (p. 192) you can obtain a list of the available parameters and their IDs. |
| Format: | SEP? [{<ItemID> <PamID>}] |
| Arguments: | <ItemID> is the item for which a parameter value from nonvolatile memory is to be queried. 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 given parameter for the given item |
| Troubleshooting: | Illegal item identifier, wrong parameter ID |

| | |
|---|---|
| Notes: | With the C-863, you can query either all parameters or one single parameter per SEP? command. |
| Available item IDs and parameter IDs: | An item is an axis, the identifier can be changed with SAI (p. 220). For further information see "Commandable Items" (p. 18).<br><br>Valid parameter IDs are given in "Parameter Overview" (p. 272). |

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

| | |
|---|---|
| Description: | Sets control value directly to move the axis. Profile generator (if present), sensor feedback and servo algorithm are not taken into account.<br><br>Servo mode must be switched off when using this command (open-loop operation). |
| Format: | SMO {<AxisID> <ControlValue>} |
| Arguments | <AxisID> is one axis of the controller.<br><br><ControlValue> is the new control value (dimensionless). See below for details. |
| Response: | None |
| Troubleshooting: | Illegal axis identifier<br><br>Servo mode is switched on for one of the specified axes. |

| | |
|---|---|
| Notes: | **NOTE: In the case of large control values, the stage can strike the hard stop despite the limit switch function. This can cause damage to equipment.**<br><br>The unsigned control value must not be larger than the value of the ***Maximum Motor Output*** parameter (0x9). When this parameter is set to its maximum (32767), <ControlValue> ranges from -32766 to 32766 (dimensionless). <ControlValue> controls the PWM signal for the axis. The sign of the value determines the direction of motion: -32766 corresponds to the maximum output voltage in negative direction of motion and 32766 to the maximum output voltage in positive direction of motion.<br><br>The ***Range Limit Min*** (0x07000000) and ***Range Limit Max*** (0x07000001) parameters can be used as soft limits for motions in open-loop operation with SMO: When the current position reaches these values, the control value is set to zero and the motion is stopped. The axis can move again as soon as the value for the soft limit has been decreased or increased. |
| Example: | Send: `SMO 1 -16000`<br>Note: The control value is about half the maximum control value. The axis moves in negative direction. |

### SMO? (Get Control Value)

| | |
|---|---|
| Description: | Gets last valid control value of given axis. |
| Format: | SMO? [{<AxisID>}] |
| Arguments | <AxisID> is one axis of the controller |
| Response: | {<AxisID>"="<float> LF}<br><br>where<br><br><float> is the last valid control value (dimensionless). For details see below. |

| | |
|---|---|
| Troubleshooting: | Illegal axis identifier |
| Notes: | The control value which is returned by SMO? can be the result of the servo algorithm and other corrections, or it can be the value set by an SMO command (p. 224) in open-loop operation. See the block diagram (p. 17) for further information.<br><br>The control value ranges from -32766 to 32766 (dimensionless) and controls the PWM signal for the axis. The sign of the value determines the direction of motion: -32766 corresponds to the maximum output voltage in negative direction of motion and 32766 to the maximum output voltage in positive direction of motion. |

### SPA (Set Volatile Memory Parameters)

| | |
|---|---|
| Description: | Sets a parameter of the given item 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 item for which a parameter is changed in volatile memory. See below for details.<br><br><PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details.<br><br><PamValue> is the value to which the given parameter of the given item is set. |
| Response: | None |

Parameter changes are also lost when the parameters are restored with RPA (p. 218).

**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. 192) returns a list of the available parameters.

SEP (p. 222) writes parameter settings directly into nonvolatile memory (without changing the settings in volatile memory).

WPA (p. 245) writes parameter settings from volatile to nonvolatile memory.

RPA resets volatile memory to the value in nonvolatile memory.

| | |
|---|---|
| Troubleshooting: | Illegal item identifier, wrong parameter ID, value out of range |
| Notes: | With the C-863 you can write only one parameter per SPA command. |
| Available item IDs and parameter IDs: | An item is an axis, the identifier can be changed with SAI (p. 220). Further information see "Commandable Items" (p. 18).<br><br>Valid parameter IDs can be found in the parameter overview (p. 272). |

| | |
|---|---|
| Example 1: | Send: `SPA 1 0x1 100` |
| | Note: Set the P-term of the servo algorithm for axis 1 to 100; the parameter ID is written in hexadecimal format |
| | Send: `SPA 1 1 150` |
| | Note: Set the P-term 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 mechanical system connected. |
| | Send: `SPA 1 0x1 150` |
| | Note: The P-term is set to 150 for axis 1. The setting is made in volatile memory only. |
| | Now set the I- and D-terms in volatile memory using SPA and then test the functioning of the system. If the closed-loop system performance proves satisfactory and you want to use this system configuration as default, save the parameter settings from volatile to non-volatile memory. |
| | Send: `WPA 100` |
| | Note: This command saves the values of all parameters whose password is 100, since WPA is used without specifying a particular parameter. |

Example 3: Send: `SEP 100 LEFT 0xA 20`

Note: The maximum closed-loop velocity must be set to 20 mm/s for axis LEFT (axis was renamed with SAI). The setting is made in non-volatile memory and hence is 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 now enabled.

Send: `SPA? LEFT 0xA`

Receive: `LEFT 0xA=20.00000`

Note: Check the parameter settings in volatile memory.

**SPA? (Get Volatile Memory Parameters)**

Description: Gets the value of a parameter of a given item from volatile memory (RAM).

With HPA? (p. 192) you can obtain a list of the available parameters.

Format: SPA? [{<ItemID> <PamID>}]

Arguments: <ItemID> is the item for which a parameter is to be queried 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}<br><br>where<br><br><PamValue> is the value of the given parameter for the given item |
| Troubleshooting: | Illegal item identifier, wrong parameter ID |
| Notes: | With the C-863, you can query either all parameters or one single parameter per SPA? command. |
| Available item IDs and parameter IDs: | An item is an axis, the identifier can be changed with SAI (p. 220). For further information see "Commandable Items" (p. 18).<br><br>Valid parameter IDs are given in "Parameter Overview" (p. 272). |

## SRG? (Query Status Register Value)

|  |  |
|---|---|
| Description: | Returns register values for queried axes and registers. |
| Format: | SRG? {<AxisID> <RegisterID>} |
| Arguments: | <AxisID> is one axis of the controller.<br><br><RegisterID>: is the ID of the specified register; see below for available registers. |
| Response: | {<AxisID><RegisterID>"="<Value> LF}<br><br>where<br><br><Value> is the value of the register; see below for details. |
| Note: | This command is identical in function to #4 (p. 154) which should be preferred when the controller is performing time-consuming tasks. |

Possible register IDs and response values: <RegisterID> can be 1.

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

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Descript-ion | On-target state | Deter-mines the reference value | In motion | Servo mode on | - | - | - | Error flag |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Descript-ion | Digital in-put line 4 | Digital in-put line 3 | Digital in-put line 2 | Digital in-put line 1 | - | Positive limit switch | Ref-erence point switch | Nega-tive limit switch |

Example: Send: `SRG? 1 1`

Receive: `1 1=0x9002`

Note: The response is given in hexadecimal format. It means that axis 1 is on target, the servo mode is ON for that axis, no error occurred, the states of the digital input lines 1 to 4 are low, and axis 1 is on the positive side of the reference point switch.

**STE (Start Step And Response Measurement)**

Description: Starts performing a step and recording the step response for the given 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. 175).

The recorded data can be read with the DRR? command (p. 178).

Format: STE <AxisID> <Amplitude>

|  |  |
|---|---|
| Arguments: | <AxisID> is one axis of the controller<br><br><Amplitude> is the size of the step. See below for details. |
| Response: | None |
| Troubleshooting: | Servo mode must be switched on for the commanded axis prior to using this command (closed-loop operation).<br><br>The target position must be inside the soft limits. Use TMN? (p. 237) and TMX? (p. 237) to ask for the current valid soft limits, and MOV? (p. 212) for the current target.<br><br>Motion commands like STE are not allowed when the joystick is active for the axis. For further information see "Joystick Control" (p. 112). |
| Notes: | A "step" consists of a relative move of the specified amplitude which is performed relative to the current position. |

**STP (Stop All Axes)**

|  |  |
|---|---|
| Description: | Stops all axes abruptly. For details see the notes below.<br><br>Sets error code to 10.<br><br>This command is identical in function to #24 (p. 156). |
| Format: | STP |
| Arguments: | None |
| Response: | None |
| Troubleshooting: | Communication breakdown |

| | |
|---|---|
| Notes: | STP stops all motion caused by motion commands (e.g. MOV (p. 211), MVR (p. 213), GOH (p. 189), STE (p. 231)), commands for reference point definition (FNL (p. 184), FPL (p. 185), FRF (p. 187)) and macros (MAC (p. 205)). Also stops macro execution.<br><br>After the axes are stopped, their target positions are set to their current positions.<br><br>HLT (p. 191) in contrast to STP stops motion with given deceleration with regard to system inertia. |

**SVO (Set Servo Mode)**

| | |
|---|---|
| Description: | Sets the servo mode for given axes (open-loop or closed-loop operation). |
| Format: | SVO {<AxisID> <ServoState>} |
| Arguments: | <AxisID> is one axis of the controller<br><br><ServoState> can have the following values:<br>0 = servo mode off (open-loop operation)<br>1 = servo mode on (closed-loop operation) |
| Response: | None |
| Troubleshooting: | Illegal axis identifier |

| | |
|---|---|
| Notes: | When switching from open-loop to closed-loop operation, the target is set to the current position to avoid jumps of the mechanics. |

The current state of the servo mode determines the applicable motion commands:
Servo mode ON: Use MOV (p. 211), MVR (p. 213), GOH (p. 189), or joystick control (p. 112)
Servo mode OFF: Use SMO (p. 224)

Servo mode must be switched on before reference moves can be started with FRF (p. 187), FNL (p. 184) or FPL (p. 185).

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

Using a startup macro, you can configure the controller so that servo mode is automatically switched on upon power-on or reboot. For further information see "Start-Up Macro".

If the axis has a brake, setting the servo mode with SVO influences the activation state of the brake:

- Switching on the servo mode deactivates the brake.
- Switching off the servo mode activates the brake. If the servo mode is switched off, the brake can be activated or deactivated with BRA (p. 161). Secure the stage against unintentional motions before you deactivate the brake with BRA!

If a motion error occurs, the servo mode is switched off and the brake is activated.  For further information see "Motion Errors" (p. 93).

### SVO? (Get Servo Mode)

| | |
|---|---|
| Description: | Gets the servo mode for the axes specified.<br><br>If all arguments are omitted, gets the servo mode of all axes. |
| Format: | SVO? [{<AxisID>}] |
| Arguments: | <AxisID> is one axis of the controller. |
| Response: | {<AxisID>"="<ServoState> LF}<br><br>where<br><br><ServoState> is the current servo mode for the axis:<br>0 = servo mode off (open-loop operation)<br>1 = servo mode on (closed-loop operation) |
| Troubleshooting: | Illegal axis identifier |

### TAC? (Tell Analog Channels)

| | |
|---|---|
| Description: | Gets the number of installed analog lines. |
| Format: | TAC? |
| Arguments: | None |
| Response: | <uint> indicates the total number of analog lines (inputs and outputs). |
| Notes: | Gets the number of analog input lines located on the **I/O** socket (p. 298) of the C-863 (Input 1 to Input 4). Note that these lines can also be used for digital input. For further information see "Commandable Items" (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}<br><br>where<br><br><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. 298) of the C-863. The identifiers of the lines are 1 to 4. For further information see "Commandable Items" (p. 18).<br><br>You can record the values of the analog input lines using the DRC record option 81 (p. 175). |

**TIO? (Tell Digital I/O Lines)**

| | |
|---|---|
| Description: | Tells number of installed digital I/O lines |
| Format: | TIO? |
| Arguments: | none |
| Response: | I=<uint1><br>O=<uint2><br><br>where<br><br><uint1> is the number of digital input lines.<br><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. 174). Furthermore, you can program the Output 1 to Output 4 lines using the CTO command (p. 164) (trigger configuration) and the TRO command (p. 238) (trigger activation/deactivation).<br><br>The digital input lines reported by TIO? are Input 1 to Input 4. They can be read with DIO? (p. 175), #4 (p. 154) and SRG? (p. 230). |

All the lines are located on the **I/O** socket (p. 298) of the C-863.

### 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. |

### 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. |

### 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-863 has 2 data recorder tables with 1024 data points per table.<br><br>For more information see "Data Recorder". |

### TRO (Set Trigger Output State)

| | |
|---|---|
| Description: | Enables or disables the trigger output conditions which were set with CTO (p. 164) for the given digital output line. |
| Format: | TRO {<TrigOutID> <TrigMode>} |
| Arguments: | <TrigOutID> is a digital output line of the controller; see below for further details.<br><br><TrigMode> can have the following values:<br>0 = Trigger output disabled<br>1 = Trigger output enabled |
| 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. 298).<br><br>Do not use DIO (p. 174) on digital output lines for which the trigger output is enabled with TRO. |

### TRO? (Get Trigger Output State)

| | |
|---|---|
| Description: | Returns if the trigger output configuration made with CTO (p. 164) is enabled or disabled for the given digital output line.<br><br>If all arguments are omitted, gets state of all digital output lines. |
| Format: | TRO? [{<TrigOutID>}] |
| Arguments: | <TrigOutID> is one digital output line of the controller, see TRO (p. 238) for more information. |
| Response: | {<TrigOutID>"="<TrigMode> LF}<br><br>where<br><br><TrigMode> is the current state of the digital output line:<br>0 = Trigger output disabled<br>1 = Trigger output enabled |
| Troubleshooting: | Illegal identifier of the digital output line |

### TRS? (Indicate Reference Switch)

| | |
|---|---|
| Description: | Indicates whether axes have a reference point switch with direction sensing. |
| Format: | TRS? [{<AxisID>}] |
| Arguments: | <AxisID> is one axis of the controller |
| Response: | {<AxisID>"="<uint> LF}<br><br>where<br><br><uint> indicates whether the axis has a direction-sensing reference point switch (=1) or not (=0). |
| Troubleshooting: | Illegal axis identifier |

| | |
|---|---|
| Notes: | The C-863 firmware detects the presence or absence of a reference point switch via a parameter (ID 0x14). According to the value of this parameter, the C-863 enables or disables reference moves to the reference point switch (FRF command (p. 187)). Adapt the parameter value to your hardware using SPA (p. 226) or SEP (p. 222). Further information, see "Reference Point Switch Detection" (p. 31).<br><br>You can use a digital input line instead of the reference point switch as source of the reference point signal for the FRF command. Further information see "Digital Input Signals" (p. 106). |

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

| | |
|---|---|
| Description: | Returns a string with characters which can be used for axis identifiers.<br><br>Use SAI (p. 220) to change the axis identifiers and SAI? (p. 221) to ask for the current valid axis identifiers. |
| Format: | TVI? |
| Arguments: | None |
| Response: | <string> is a list of characters |
| Notes: | With the C-863, the string consists of<br>1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ-_ |

## VAR (Set Variable Value)

| | |
|---|---|
| Description: | Sets a variable to a certain value.<br><br>Local variables can be set using VAR in macros only. See “Variables” (p. 148) for details regarding local and global variables.<br><br>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.<br><br><String> is the value to which the variable is to be set. If omitted, the variable is deleted.<br><br>The value can be given directly or via the value of a variable.<br><br>See “Variables” (p. 148) for conventions regarding variable names and values. |
| Response: | None |
| Example: | It is possible to set the value of one variable (e.g. TARGET) to that of another variable (e.g. SOURCE): |

```
VAR TARGET ${SOURCE}
```

Use curly brackets if the name of the variable is longer than one character:

```
VAR A ONE
VAR VARB TWO
VAR $A 1
VAR ${VARB} 2
VAR $VARB 2 // this will result in an unwanted behavior
VAR?
A=ONE
VARB=TWO
ONE=1
TWO=2 // ${VARB}: is replaced by its value “TWO”.
ARB=2 // $VARB: $V is replaced by its (empty) value.
```

See ADD (p. 158) for another example.

### VAR? (Get Variable Values)

| | |
|---|---|
| Description: | Gets values of variables.<br><br>If VAR? is combined with CPY (p. 163), JRC (p. 202), MEX (p. 209) or WAC (p. 244), the response to VAR? has to be a single value and not more.<br><br>More information regarding local and global variables can be found in “Variables” (p. 148). |
| Format: | VAR? [{<Variable>}] |
| Arguments: | <Variable> is the name of the variable to be queried. More information on name conventions can be found in "Variables" (p. 148).<br><br>If <Variable> is omitted, all global variables present in the RAM are listed. |
| Response: | {<Variable>”=”<String>LF}<br><br>where<br><br><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. 148) for details regarding local and global variables. |
| Example: | See ADD (p. 158) for an example. |

### VEL (Set Closed-Loop Velocity)

| | |
|---|---|
| Description: | Sets velocity of given axes.<br><br>The velocity can be changed with VEL while the axis is moving. |
| Format: | VEL {<AxisID> <Velocity>} |
| Arguments: | <AxisID> is one axis of the controller.<br><br><Velocity> is the velocity value in physical units/s. |

| | |
|---|---|
| Response: | None |
| Troubleshooting: | Illegal axis identifiers |
| Notes: | The VEL setting only takes effect when the given axis is in closed-loop operation (servo mode ON).<br><br>The lowest possible value for <Velocity> is 0.<br><br>VEL changes the value of the ***Closed-Loop Velocity (Phys. Unit/s)*** parameter (ID 0x49) in the volatile memory (can be saved as default with WPA, can also be changed with SPA (p. 226) and SEP (p. 222)).<br><br>The maximum value which can be set with the VEL command is given by the ***Maximum Closed-Loop Velocity (Phys. Unit/s)*** parameter, ID 0xA (can be changed with SPA (p. 226) and SEP (p. 222)). |

**VEL? (Get Closed-Loop Velocity)**

| | |
|---|---|
| Description: | Gets the velocity value commanded with VEL (p. 242).<br><br>If all arguments are omitted, the value commanded with VEL is queried for all axes. |
| Format: | VEL? [{<AxisID>}] |
| Arguments: | <AxisID> is one axis of the controller. |
| Response: | {<AxisID>"="<float> LF}<br><br>where<br><br><float> is the velocity value commanded with VEL in physical units/s. |
| Notes: | VEL? queries the velocity value for closed-loop operation. |

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

| | |
|---|---|
| Description: | Gets the versions of the firmware of the C-863 as well as of further components like, for example, drivers and libraries. |
| Format: | VER? |
| Arguments: | None |
| Response | {<string1>":" <string2> [<string3>]LF}<br><br>where<br><br><string1> is the name of the component;<br><string2> is the version information of the component <string1>;<br><string3> is an optional note. |

### WAC (Wait For Condition)

| | |
|---|---|
| Description: | Waits until a given condition of the following type occurs: a specified value is compared with a queried value according a specified rule.<br><br>Can only be used in macros.<br><br>See also the MEX command (p. 209). |
| 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.<br><br><OP> is the operator to be used. The following operators are possible:<br>= <= < > >= !=<br>Important: There must be a blank space before and after the operator!<br><br><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 Nonvolatile Memory)

Description: Writes the currently valid value of a parameter of a given item 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. 218) is used to restore the parameters.

With HPA? (p. 192) you can obtain a list of all available parameters.

Use SPA? (p. 226) to check the current parameter settings in volatile memory.

See SPA (p. 226) for an example.

| | |
|---|---|
| Format: | WPA <Pswd> [{<ItemID> <PamID>}] |
| Arguments: | <Pswd> is the password for writing to the nonvolatile memory. See below for details.<br><br><ItemID> is the item for which a parameter is to be saved from the volatile to the nonvolatile memory. See below for details.<br><PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details. |
| Response: | None |
| Troubleshooting: | Illegal item identifier, wrong parameter ID, invalid password |
| Notes: | Parameters can be changed in volatile memory with SPA (p. 226), ACC (p. 157), DEC (p. 170) and VEL (p. 242).<br><br>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.<br><br>**Note that the number of write cycles in the nonvolatile memory is limited. Write default settings only if necessary.** |
| Valid passwords: | The password for writing to non-volatile memory is "100". |
| Available item IDs and parameter IDs: | An item is an axis, the identifier can be changed with SAI (p. 220). For further information see "Commandable Items" (p. 18).<br><br>Valid parameter IDs are given in "Parameter Overview" (p. 272). |

## 9.7 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 cannot 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 nonvolatile 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 cannot be started |
| 69 | PI_CNTR_DDL_TIME_DELAY_TOO_LARGE | Time delay larger than DDL table; DDL cannot 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 overran and command couldn't be received correctly |
| 77 | PI_CNTR_NOT_ENOUGH_RECORDED_DATA | Data record table does not hold enough recorded data |
| 78 | PI_CNTR_TABLE_DEACTIVATED | Data record table is not configured for recording |
| 79 | PI_CNTR_OPENLOOP_VALUE_SET_WHEN_SERVO_ON | Open-loop commands (SVA, SVR) are not allowed when servo is on |
| 80 | PI_CNTR_RAM_ERROR | Hardware error affecting RAM |
| 81 | PI_CNTR_MACRO_UNKNOWN_COMMAND | Not macro command |
| 82 | PI_CNTR_MACRO_PC_ERROR | Macro counter out of range |
| 83 | PI_CNTR_JOYSTICK_ACTIVE | Joystick is active |
| 84 | PI_CNTR_MOTOR_IS_OFF | Motor is off |
| 85 | PI_CNTR_ONLY_IN_MACRO | Macro-only command |
| 86 | PI_CNTR_JOYSTICK_UNKNOWN_AXIS | Invalid joystick axis |
| 87 | PI_CNTR_JOYSTICK_UNKNOWN_ID | Joystick unknown |
| 88 | PI_CNTR_REF_MODE_IS_ON | Move without referenced stage |
| 89 | PI_CNTR_NOT_ALLOWED_IN_CURRENT_MOTION_MODE | Command not allowed in current motion mode |
| 90 | PI_CNTR_DIO_AND_TRACING_NOT_POSSIBLE | No tracing possible while digital IOs are used on this HW revision. Reconnect to switch operation mode. |
| 91 | PI_CNTR_COLLISION | Move not possible, would cause collision |
| 92 | PI_CNTR_SLAVE_NOT_FAST_ENOUGH | Stage is not capable of following the master. Check the gear ratio. |
| 93 | PI_CNTR_CMD_NOT_ALLOWED_WHILE_AXIS_IN_MOTION | This command is not allowed while the affected axis or its master is in motion. |
| 94 | PI_CNTR_OPEN_LOOP_JOYSTICK_ENABLED | Servo cannot be switched on when open-loop joystick control is enabled. |
| 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 |
| 100 | PI_LABVIEW_ERROR | PI LabVIEW driver reports error. See source control for details. |

| | | |
|---|---|---|
| 200 | PI_CNTR_NO_AXIS | No stage connected to axis |
| 201 | PI_CNTR_NO_AXIS_PARAM_FILE | File with axis parameters not found |
| 202 | PI_CNTR_INVALID_AXIS_PARAM_FILE | Invalid axis parameter file |
| 203 | PI_CNTR_NO_AXIS_PARAM_BACKUP | Backup file with axis parameters not found |
| 204 | PI_CNTR_RESERVED_204 | PI internal error code 204 |
| 205 | PI_CNTR_SMO_WITH_SERVO_ON | SMO with servo on |
| 206 | PI_CNTR_UUDECODE_INCOMPLETE_HEADER | uudecode: incomplete header |
| 207 | PI_CNTR_UUDECODE_NOTHING_TO_DECODE | uudecode: nothing to decode |
| 208 | PI_CNTR_UUDECODE_ILLEGAL_FORMAT | uudecode: illegal UUE format |
| 209 | PI_CNTR_CRC32_ERROR | CRC32 error |
| 210 | PI_CNTR_ILLEGAL_FILENAME | Illegal file name (must be 8-0 format) |
| 211 | PI_CNTR_FILE_NOT_FOUND | File not found on controller |
| 212 | PI_CNTR_FILE_WRITE_ERROR | Error writing file on controller |
| 213 | PI_CNTR_DTR_HINDERS_VELOCITY_CHANGE | VEL command not allowed in DTR command mode |
| 214 | PI_CNTR_POSITION_UNKNOWN | Position calculations failed |
| 215 | PI_CNTR_CONN_POSSIBLY_BROKEN | The connection between controller and stage may be broken |
| 216 | PI_CNTR_ON_LIMIT_SWITCH | The connected stage has driven into a limit switch, some controllers need CLR to resume operation |
| 217 | PI_CNTR_UNEXPECTED_STRUT_STOP | Strut test command failed because of an unexpected strut stop |
| 218 | PI_CNTR_POSITION_BASED_ON_ESTIMATION | While MOV! is running position can only be estimated! |
| 219 | PI_CNTR_POSITION_BASED_ON_INTERPOLATION | Position was calculated during MOV motion |
| 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. |
| 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 |
| 534 | PI_CNTR_NODE_CHAIN_INVALID | The node chain does not end in the \"0\" node |
| 535 | PI_CNTR_NODE_DEFINITION_NOT_CONSISTENT | The definition of a coordinate transformation is erroneous |
| 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 |
| 538 | PI_CNTR_NODE_TYPE_DIFFERS | A node can only be replaced by a node of the same type |
| 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) |
| 541 | PI_CNTR_ZERO_NODE_CANNOT_BE_CHANGED_OR_REPLACED | 0 is the root node and cannot be modified |
| 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 |
| 555 | PI_CNTR_UNKNOWN_ERROR | BasMac: unknown controller error |
| 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 |
| 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 answer was received while executing WAC/MEX/JRC/... |
| 1011 | PI_CMD_NOT_VALID_IN_MACRO_MODE | Command not valid during macro execution |
| 1024 | PI_CNTR_MOTION_ERROR | Motion error: position error too large, servo is switched off automatically |
| 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 |
| 5000 | PI_CNTR_INVALID_PCC_SCAN_DATA | PicoCompensation scan data is not valid |
| 5001 | PI_CNTR_PCC_SCAN_RUNNING | PicoCompensation is running, some actions cannot be executed during scanning/recording |
| 5002 | PI_CNTR_INVALID_PCC_AXIS | Given axis cannot be defined as PPC axis |
| 5003 | PI_CNTR_PCC_SCAN_OUT_OF_RANGE | Defined scan area is larger than the travel range |
| 5004 | PI_CNTR_PCC_TYPE_NOT_EXISTING | Given PicoCompensation type is not defined |
| 5005 | PI_CNTR_PCC_PAM_ERROR | PicoCompensation parameter error |
| 5006 | PI_CNTR_PCC_TABLE_ARRAY_TOO_LARGE | PicoCompensation table is larger than maximum table length |
| 5100 | PI_CNTR_NEXLINE_ERROR | Common error in NEXLINE® firmware module |
| 5101 | PI_CNTR_CHANNEL_ALREADY_USED | Output channel for NEXLINE® cannot 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 cannot 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 |
| 6000 | PI_CNTR_SENSOR_ABS_INVALID_VALUE | Invalid preset value of absolute sensor |
| 6001 | PI_CNTR_SENSOR_ABS_WRITE_ERROR | Error while writing to sensor |
| 6002 | PI_CNTR_SENSOR_ABS_READ_ERROR | Error while reading from sensor |
| 6003 | PI_CNTR_SENSOR_ABS_CRC_ERROR | Checksum error of absolute sensor |
| 6004 | PI_CNTR_SENSOR_ABS_ERROR | General error of absolute sensor |
| 6005 | PI_CNTR_SENSOR_ABS_OVERFLOW | Overflow of absolute sensor position |

## Interface errors

| | | |
|---|---|---|
| 0 | COM_NO_ERROR | No error occurred during function call |
| -1 | COM_ERROR | Error during com operation (could not be specified) |
| -2 | SEND_ERROR | Error while sending data |
| -3 | REC_ERROR | Error while receiving data |
| -4 | NOT_CONNECTED_ERROR | Not connected (no port with given ID open) |
| -5 | COM_BUFFER_OVERFLOW | Buffer overflow |
| -6 | CONNECTION_FAILED | Error while opening port |
| -7 | COM_TIMEOUT | Timeout error |
| -8 | COM_MULTILINE_RESPONSE | There are more lines waiting in buffer |
| -9 | COM_INVALID_ID | There is no interface or DLL handle with the given ID |
| -10 | COM_NOTIFY_EVENT_ERROR | Event/message for notification could not be opened |
| -11 | COM_NOT_IMPLEMENTED | Function not supported by this interface type |
| -12 | COM_ECHO_ERROR | Error while sending "echoed" data |
| -13 | COM_GPIB_EDVR | IEEE488: System error |
| -14 | COM_GPIB_ECIC | IEEE488: Function requires GPIB board to be CIC |
| -15 | COM_GPIB_ENOL | IEEE488: Write function detected no listeners |
| -16 | COM_GPIB_EADR | IEEE488: Interface board not addressed correctly |
| -17 | COM_GPIB_EARG | IEEE488: Invalid argument to function call |
| -18 | COM_GPIB_ESAC | IEEE488: Function requires GPIB board to be SAC |
| -19 | COM_GPIB_EABO | IEEE488: I/O operation aborted |
| -20 | COM_GPIB_ENEB | IEEE488: Interface board not found |
| -21 | COM_GPIB_EDMA | IEEE488: Error performing DMA |
| -22 | COM_GPIB_EOIP | IEEE488: I/O operation started before previous operation completed |
| -23 | COM_GPIB_ECAP | IEEE488: No capability for intended operation |
| -24 | COM_GPIB_EFSO | IEEE488: File system operation error |

| | | |
|---|---|---|
| -25 | COM_GPIB_EBUS | IEEE488: Command error during device call |
| -26 | COM_GPIB_ESTB | IEEE488: Serial poll-status byte lost |
| -27 | COM_GPIB_ESRQ | IEEE488: SRQ remains asserted |
| -28 | COM_GPIB_ETAB | IEEE488: Return buffer full |
| -29 | COM_GPIB_ELCK | IEEE488: Address or board locked |
| -30 | COM_RS_INVALID_DATA_BITS | RS-232: 5 data bits with 2 stop bits is an invalid combination, as is 6, 7, or 8 data bits with 1.5 stop bits |
| -31 | COM_ERROR_RS_SETTINGS | RS-232: Error configuring the COM port |
| -32 | COM_INTERNAL_RESOURCES_ERROR | Error dealing with internal system resources (events, threads, ...) |
| -33 | COM_DLL_FUNC_ERROR | A DLL or one of the required functions could not be loaded |
| -34 | COM_FTDIUSB_INVALID_HANDLE | FTDIUSB: invalid handle |
| -35 | COM_FTDIUSB_DEVICE_NOT_FOUND | FTDIUSB: device not found |
| -36 | COM_FTDIUSB_DEVICE_NOT_OPENED | FTDIUSB: device not opened |
| -37 | COM_FTDIUSB_IO_ERROR | FTDIUSB: IO error |
| -38 | COM_FTDIUSB_INSUFFICIENT_RESOURCES | FTDIUSB: insufficient resources |
| -39 | COM_FTDIUSB_INVALID_PARAMETER | FTDIUSB: invalid parameter |
| -40 | COM_FTDIUSB_INVALID_BAUD_RATE | FTDIUSB: invalid baud rate |
| -41 | COM_FTDIUSB_DEVICE_NOT_OPENED_FOR_ERASE | FTDIUSB: device not opened for erase |
| -42 | COM_FTDIUSB_DEVICE_NOT_OPENED_FOR_WRITE | FTDIUSB: device not opened for write |
| -43 | COM_FTDIUSB_FAILED_TO_WRITE_DEVICE | FTDIUSB: failed to write 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!) |

### DLL errors

| | | |
|---|---|---|
| -1001 | PI_UNKNOWN_AXIS_IDENTIFIER | Unknown axis identifier |
| -1002 | PI_NR_NAV_OUT_OF_RANGE | Number for NAV out of range--must be in [1,10000] |
| -1003 | PI_INVALID_SGA | Invalid value for SGA--must be one of 1, 10, 100, 1000 |
| -1004 | PI_UNEXPECTED_RESPONSE | Controller sent unexpected response |
| -1005 | PI_NO_MANUAL_PAD | No manual control pad installed, calls to SMA and related commands are not allowed |
| -1006 | PI_INVALID_MANUAL_PAD_KNOB | Invalid number for manual control pad knob |
| -1007 | PI_INVALID_MANUAL_PAD_AXIS | Axis not currently controlled by a manual control pad |
| -1008 | PI_CONTROLLER_BUSY | Controller is busy with some lengthy operation (e.g. reference move, fast scan algorithm) |
| -1009 | PI_THREAD_ERROR | Internal error--could not start thread |
| -1010 | PI_IN_MACRO_MODE | Controller is (already) in macro mode--command not valid in macro mode |
| -1011 | PI_NOT_IN_MACRO_MODE | Controller not in macro mode--command not valid unless macro mode active |
| -1012 | PI_MACRO_FILE_ERROR | Could not open file to write or read macro |
| -1013 | PI_NO_MACRO_OR_EMPTY | No macro with given name on controller, or macro is empty |
| -1014 | PI_MACRO_EDITOR_ERROR | Internal error in macro editor |

| | | |
|---|---|---|
| -1015 | PI_INVALID_ARGUMENT | One or more arguments given to function is invalid (empty string, index out of range, ...) |
| -1016 | PI_AXIS_ALREADY_EXISTS | Axis identifier is already in use by a connected stage |
| -1017 | PI_INVALID_AXIS_IDENTIFIER | Invalid axis identifier |
| -1018 | PI_COM_ARRAY_ERROR | Could not access array data in COM server |
| -1019 | PI_COM_ARRAY_RANGE_ERROR | Range of array does not fit the number of parameters |
| -1020 | PI_INVALID_SPA_CMD_ID | Invalid parameter ID given to SPA or SPA? |
| -1021 | PI_NR_AVG_OUT_OF_RANGE | Number for AVG out of range--must be >0 |
| -1022 | PI_WAV_SAMPLES_OUT_OF_RANGE | Incorrect number of samples given to WAV |
| -1023 | PI_WAV_FAILED | Generation of wave failed |
| -1024 | PI_MOTION_ERROR | Motion error: position error too large, servo is switched off automatically |
| -1025 | PI_RUNNING_MACRO | Controller is (already) running a macro |
| -1026 | PI_PZT_CONFIG_FAILED | Configuration of PZT stage or amplifier failed |
| -1027 | PI_PZT_CONFIG_INVALID_PARAMS | Current settings are not valid for desired configuration |
| -1028 | PI_UNKNOWN_CHANNEL_IDENTIFIER | Unknown channel identifier |
| -1029 | PI_WAVE_PARAM_FILE_ERROR | Error while reading/writing wave generator parameter file |
| -1030 | PI_UNKNOWN_WAVE_SET | Could not find description of wave form. Maybe WG.INI is missing? |
| -1031 | PI_WAVE_EDITOR_FUNC_NOT_LOADED | The WGWaveEditor DLL function was not found at startup |
| -1032 | PI_USER_CANCELLED | The user cancelled a dialog |
| -1033 | PI_C844_ERROR | Error from C-844 Controller |
| -1034 | PI_DLL_NOT_LOADED | DLL necessary to call function not loaded, or function not found in DLL |
| -1035 | PI_PARAMETER_FILE_PROTECTED | The open parameter file is protected and cannot be edited |
| -1036 | PI_NO_PARAMETER_FILE_OPENED | There is no parameter file open |
| -1037 | PI_STAGE_DOES_NOT_EXIST | Selected stage does not exist |

| | | |
|---|---|---|
| -1038 | PI_PARAMETER_FILE_ALREADY_OPENED | There is already a parameter file open. Close it before opening a new file |
| -1039 | PI_PARAMETER_FILE_OPEN_ERROR | Could not open parameter file |
| -1040 | PI_INVALID_CONTROLLER_VERSION | The version of the connected controller is invalid |
| -1041 | PI_PARAM_SET_ERROR | Parameter could not be set with SPA--parameter not defined for this controller! |
| -1042 | PI_NUMBER_OF_POSSIBLE_WAVES_EXCEEDED | The maximum number of wave definitions has been exceeded |
| -1043 | PI_NUMBER_OF_POSSIBLE_GENERATORS_EXCEEDED | The maximum number of wave generators has been exceeded |
| -1044 | PI_NO_WAVE_FOR_AXIS_DEFINED | No wave defined for specified axis |
| -1045 | PI_CANT_STOP_OR_START_WAV | Wave output to axis already stopped/started |
| -1046 | PI_REFERENCE_ERROR | Not all axes could be referenced |
| -1047 | PI_REQUIRED_WAVE_NOT_FOUND | Could not find parameter set required by frequency relation |
| -1048 | PI_INVALID_SPP_CMD_ID | Command ID given to SPP or SPP? is not valid |
| -1049 | PI_STAGE_NAME_ISNT_UNIQUE | A stage name given to CST is not unique |
| -1050 | PI_FILE_TRANSFER_BEGIN_MISSING | A uuencoded file transferred did not start with "begin" followed by the proper filename |
| -1051 | PI_FILE_TRANSFER_ERROR_TEMP_FILE | Could not create/read file on host PC |
| -1052 | PI_FILE_TRANSFER_CRC_ERROR | Checksum error when transferring a file to/from the controller |
| -1053 | PI_COULDNT_FIND_PISTAGES_DAT | The PiStages.dat database could not be found. This file is required to connect a stage with the CST command |
| -1054 | PI_NO_WAVE_RUNNING | No wave being output to specified axis |
| -1055 | PI_INVALID_PASSWORD | Invalid password |
| -1056 | PI_OPM_COM_ERROR | Error during communication with OPM (Optical Power Meter), maybe no OPM connected |
| -1057 | PI_WAVE_EDITOR_WRONG_PARAMNUM | WaveEditor: Error during wave creation, incorrect number of parameters |
| -1058 | PI_WAVE_EDITOR_FREQUENCY_OUT_OF_RANGE | WaveEditor: Frequency out of range |
| -1059 | PI_WAVE_EDITOR_WRONG_IP_VALUE | WaveEditor: Error during wave creation, incorrect index for integer parameter |

| | | |
|---|---|---|
| -1060 | PI_WAVE_EDITOR_WRONG_DP_VALUE | WaveEditor: Error during wave creation, incorrect index for floating point parameter |
| -1061 | PI_WAVE_EDITOR_WRONG_ITEM_VALUE | WaveEditor: Error during wave creation, could not calculate value |
| -1062 | PI_WAVE_EDITOR_MISSING_GRAPH_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 see the documentation to determine the required device driver version. |
| -1075 | PI_INVALID_LIBRARY_VERSION | The library used doesn't match the required version. Please see the documentation to determine the required library version. |
| -1076 | PI_INTERFACE_LOCKED | The interface is currently locked by another function. Please try again later. |

| | | |
|---|---|---|
| -1077 | PI_PARAM_DAT_FILE_INVALID_VERSION | Version of parameter DAT file does not match the required version. Current files are available at www.pi.ws. |
| -1078 | PI_CANNOT_WRITE_TO_PARAM_DAT_FILE | Cannot write to parameter DAT file to store user defined stage type. |
| -1079 | PI_CANNOT_CREATE_PARAM_DAT_FILE | Cannot create parameter DAT file to store user defined stage type. |
| -1080 | PI_PARAM_DAT_FILE_INVALID_REVISION | Parameter DAT file does not have correct revision. |
| -1081 | PI_USERSTAGES_DAT_FILE_INVALID_REVISION | User stages DAT file does not have correct revision. |
| -1082 | PI_SOFTWARE_TIMEOUT | Timeout Error. Some lengthy operation did not finish within expected time. |

# 10 Adapting Settings

## In this Chapter

## 10.1 Changing Parameters in the C-863

***INFORMATION***

The values in the nonvolatile memory are loaded to the volatile memory as default values when the C-863 is switched on or rebooted and take effect immediately.

***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. 265) before you perform changes in the nonvolatile memory.
➢ Contact our customer service department (p. 291) if the C-863 exhibits unexpected behavior.

### 10.1.1 General Commands for Parameters

The following general commands are available for changing parameters:

| Command | Function |
|---|---|
| `SPA` | Change parameters in the volatile memory. |
| `SEP` | Change parameters in the nonvolatile memory. |
| `WPA` | Copy a current parameter value from the volatile to the nonvolatile memory. Here it is used as a default value. |

| Command | Function |
|---|---|
| `RPA` | Copy a parameter value from the nonvolatile to the volatile memory. |
| `SPA?` | Get parameter values from the volatile memory. |
| `SEP?` | Get parameter values from the nonvolatile memory. |

You can find details in the command descriptions (p. 154).

You get easier access to the parameter values with the PIMikroMove PC software.

### 10.1.2 Special Commands for Parameters

The following special commands only change the corresponding parameters in the volatile memory. When necessary, the changed values must be written to the nonvolatile memory with the `WPA` command (p. 245).

***INFORMATION***

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

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

You can find details in the command descriptions (p. 154).

## 10.1.3 Saving Parameter Values in a Text File

***INFORMATION***

The C-863 is configured via parameters, e. g. for adaptation to the connected mechanical system. Changing parameter values can cause undesirable results.

- ➢ Create a backup copy on the PC before changing the parameter settings of the C-863. You can then restore the original settings at any time.
- ➢ Create an additional backup copy with a new filename each time after you optimize the parameter values or adapt the C-863 to a particular stage.

### Prerequisites

- ✓ You have read and understood the General Notes on Start-Up (p. 67).
- ✓ You have established communication between the C-863 and the PC with PIMikroMove or PITerminal via the RS-232 interface (p. 72) or the USB interface (p. 74).

### Saving parameter values in a text file

1. If you use PIMikroMove, open the window for sending commands:
   - In the main window select the ***Tools > Command entry*** menu item 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-863: Send the `SPA?` command.
   - If you want to save the parameter values from the nonvolatile memory of the C-863: Send the `SEP?` command.
3. Click on the ***Save...*** button.

   The ***Save content of terminal as textfile*** window opens.
4. In the ***Save content of terminal as textfile*** window, save the queried parameter values in a text file on your PC.

### 10.1.4 Changing Parameters: General Procedure

*NOTICE*

**Improper parameter settings!**

The values in the nonvolatile memory are loaded to the volatile memory as default values when switching on or rebooting the C-863 and take effect immediately.

Improper parameter settings can cause damage to the stage connected.

➢ Only change parameters after careful consideration.

➢ Save the current parameter values to the PC (p. 265) before you perform changes in the nonvolatile memory.

#### Available parameters

The parameters available for adapting the C-863 to your application depend on the firmware of your C-863.

➢ Send the `HPA?` command (p. 192) to obtain a list of all available parameters with a short description.

#### Parameters in the nonvolatile memory

➢ Send the `SEP?` command (p. 223) to obtain a list of parameter values in the nonvolatile memory.

*INFORMATION*

The parameter values in the nonvolatile memory are automatically loaded to the volatile memory when switching on or rebooting the C-863.

➢ Change the parameters in the nonvolatile memory only if you are sure that the C-863 functions correctly with the parameter values.

➢ Change the parameters in the nonvolatile memory with the `SEP` command (p. 222).

#### Parameters in the volatile memory

➢ Send the `SPA?` command (p. 229) to obtain a list of parameter values in the volatile memory.

➢ Change the parameters in the volatile memory with the `SPA` command (p. 226).

- and/or -

➢ Change selected parameters in the volatile memory with special commands (p. 264).

If you are working with PIMikroMove:

1. In the main window of PIMikroMove, open the single axis window for the connected stage by selecting the stage in the ***View > Single Axis Window*** menu.
2. Expand the view of the single axis window by clicking on the **>** button at the right edge of the window.
3. If the parameter to be modified is not included in the list on the right side of the window, click on ***Select parameters...*** and add it to the list.
4. Type the new parameter value into the appropriate input field in the list.
5. Press the Enter key of the PC to transfer the parameter value to the volatile memory of the controller. When this is done, the entry changes its color from blue to black.

## Writing parameters from the volatile memory to the non-volatile memory

1. Change the parameters in the volatile memory with the `SPA` command (p. 226).
2. Check whether the C-863 is operating correctly with the modified parameters.
   - If yes:
     Write the modified parameter values to the non-volatile memory with the `WPA` command (p. 245).
   - If no:
     Change and check the parameters in the volatile memory again.

If you are working with PIMikroMove:

➢ Write the updated values of all parameters to the non-volatile memory:

a) In the main window of PIMikroMove, select the ***C-863 > Save parameters to non-volatile memory*** menu item.
b) When prompted, enter the password 100.

### Writing parameters from the nonvolatile memory to the volatile memory

***INFORMATION***

➢ Use this procedure only if you are sure that the C-863 functions correctly with the parameter values from the nonvolatile memory.

➢ Write the parameter values from the nonvolatile memory to the volatile memory with the `RPA` command (p. 218).

## 10.2 Creating or Modifying a Stage Type

You can select a parameter set appropriate for your stage from a stage database in the PC software from PI. The software transfers the values of the selected parameter set to the volatile memory of the controller. Further information see "Stage Databases" (p. 44).

The PI_UserStages2.dat stage database is intended for creating, editing and storing new parameter sets. This can be required in the following cases, for example:

- You want to operate a stage with different servo-control parameters than those of the PIStages2.dat stage database.
- You want to adapt the soft limits of the stage to your application.
- You have a custom stage.

***INFORMATION***

You can create a new stage type with utmost ease by modifying in PIMikroMove an existing stage type and saving it under a new name.

***INFORMATION***

If a stage type with the same name is present in the PIStages2.dat and PI_UserStages2.dat stage databases, the parameter settings from PIStages2.dat are always loaded when selecting the stage type in the PC software. The parameter settings from PI_UserStages2.dat are not used in this case.

➢ When saving stage types only assign names that are **not** already used in the PIStages2.dat stage database.

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

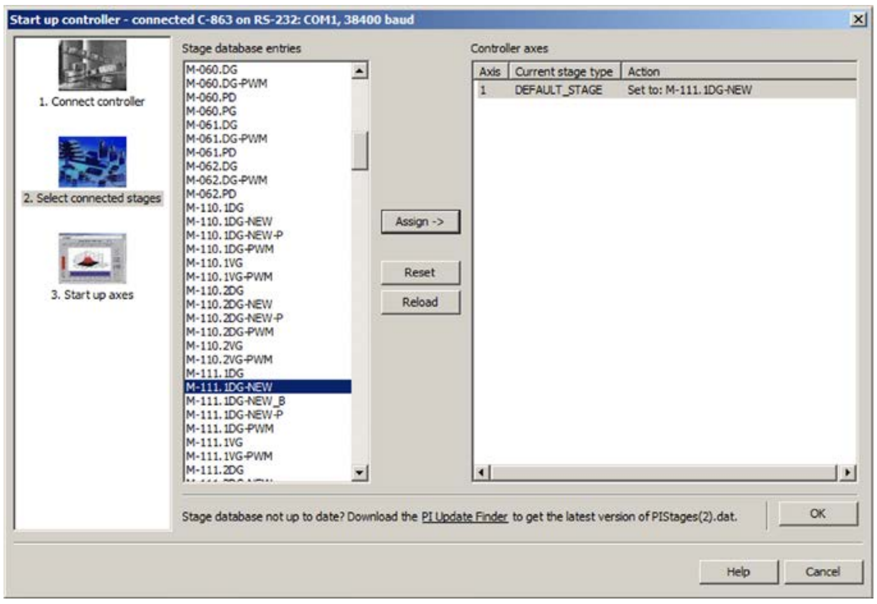
## Prerequisite

- ✓ PIMikroMove is installed on the PC (p. 53).
- ✓ You have read and understood the PIMikroMove manual. The manual is found on the product CD.
- ✓ You have installed the latest version of the PIStages2.dat stage database on the PC (p. 54).
- ✓ If PI has provided you with a custom stage database for your stage, then you have installed this database on your PC (p. 56).
- ✓ You have established communication (p. 72) between the C-863 and the PC with PIMikroMove.

## Creating a stage type in a stage database

1. Select an appropriate stage type in PIMikroMove in the ***Select connected stages*** step:

   a) Mark the stage type in the ***Stage database entries*** list.
   b) Click ***Assign***.
   c) Confirm the selection with ***OK*** to load the parameter settings for the selected stage type from the stage database to the volatile memory of the C-863.


Start up controller - connected C-863 on RS-232: COM1, 38400 baud

Stage database entries:
M-060.DG
M-060.DG-PWM
M-060.PD
M-060.PG
M-061.DG
M-061.DG-PWM
M-061.PD
M-062.DG
M-062.DG-PWM
M-062.PD
M-110.1DG
M-110.1DG-NEW
M-110.1DG-NEW-P
M-110.1DG-PWM
M-110.1VG
M-110.1VG-PWM
M-110.2DG
M-110.2DG-NEW
M-110.2DG-NEW-P
M-110.2DG-PWM
M-110.2VG
M-110.2VG-PWM
M-111.1DG
M-111.1DG-NEW
M-111.1DG-NEW_B
M-111.1DG-NEW-P
M-111.1DG-PWM
M-111.1VG
M-111.1VG-PWM
M-111.2DG

Controller axes

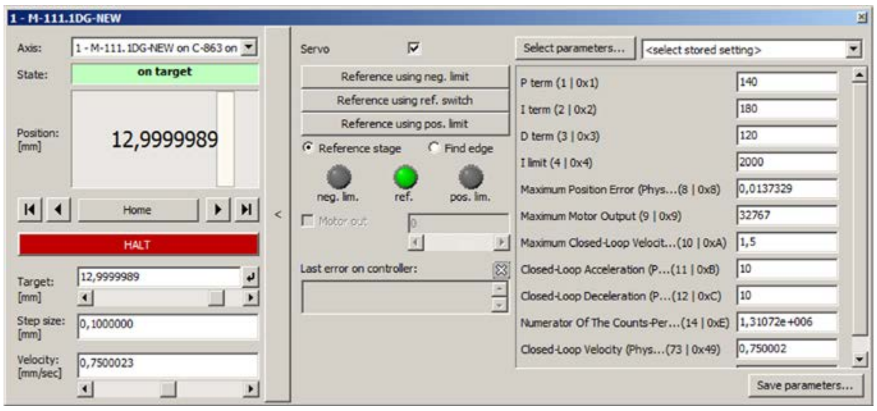
| Axis | Current stage type | Action |
|---|---|---|
| 1 | DEFAULT_STAGE | Set to: M-111.1DG-NEW |

Stage database not up to date? Download the PI Update Finder to get the latest version of PIStages(2).dat.

*Figure 33: Start up controller – Select connected stages*

After clicking ***OK*** the ***Start up controller*** window switches to the ***Start up axes*** step.

2. In the ***Start up axes*** step, click on ***Close*** to close the ***Start up controller*** window.
3. In the main window of PIMikroMove, open the single axis window for the stage type selected by selecting the stage type in the ***View > Single Axis Window*** menu.
4. Expand the view of the single axis window by clicking on the **>** button at the right edge of the window.


1 - M-111.1DG-NEW

Axis: 1 - M-111.1DG-NEW on C-863 on
State: on target
Position: [mm] 12,9999989
Target: [mm] 12,9999989
Step size: [mm] 0,1000000
Velocity: [mm/sec] 0,7500023

Servo ☑
Reference stage [✓] / Find edge
Motor out ☐ 0
Last error on controller:

Select parameters... <select stored setting>

| | |
|---|---|
| P term (1 \| 0x1) | 140 |
| I term (2 \| 0x2) | 180 |
| D term (3 \| 0x3) | 120 |
| I limit (4 \| 0x4) | 2000 |
| Maximum Position Error (Phys...(8 \| 0x8) | 0,0137329 |
| Maximum Motor Output (9 \| 0x9) | 32767 |
| Maximum Closed-Loop Velocit...(10 \| 0xA) | 1,5 |
| Closed-Loop Acceleration (P...(11 \| 0xB) | 10 |
| Closed-Loop Deceleration (P...(12 \| 0xC) | 10 |
| Numerator Of The Counts-Per...(14 \| 0xE) | 1,31072e+006 |
| Closed-Loop Velocity (Phys...(73 \| 0x49) | 0,750002 |

Home
HALT
Reference using neg. limit
Reference using ref. switch
Reference using pos. limit
neg. lim.
ref.
pos. lim.
<
Save parameters...

*Figure 34: PIMikroMove: Expanded single axis window*

5. Enter new values for the stage parameters:
   a) If the parameter to be modified is not included in the list on the right side of the window, click on ***Select parameters...*** and add it to the list.
   b) Type the new parameter value into the appropriate input field in the list.
   c) Press the Enter key of the PC to transfer the parameter value to the volatile memory of the controller. When this is done, the entry changes its color from blue to black.
6. Click the ***Save parameters…*** button below the parameter list.

   The ***Save Parameters as User Stage Type*** dialog opens.
7. In the ***Save Parameters as User Stage Type*** dialog, save the modified parameter values as a new stage type:
   a) Leave the entry in the ***Parameters of axis:*** field unchanged.
   b) Enter the name for the new stage type in the***Save as:*** field.
   c) Click on ***OK***.

The new stage type has been saved in the PI_UserStages2.dat stage database. The display of the stage type connected has been updated in the single axis window and in the main window of PIMikroMove. The new stage type is available with immediate effect for selection in the ***Select connected stages*** step too.

### Modifying a stage type in a stage database

1. In the ***Select connected stages*** step in PIMikroMove, select a stage type which you created earlier as described above. Proceed with the selection as in step 1 of the **Creating a stage type in a stage database** instruction.
2. Execute steps 2 to 6 from **Creating a stage type in a stage database**.
3. In the ***Save Parameters as User Stage Type*** dialog, save the modified parameter values as a new stage type:

   a) Leave the entry in the ***Parameters of axis:*** field unchanged.
   b) Leave the entry in the ***Save as:*** field unchanged.
   c) Click on ***OK***.
   d) In the ***Stage type already defined*** dialog, click on ***Change settings***. The ***Save Parameters as User Stage Type*** dialog closes automatically after a short time.

   The parameter values of the stage type have been updated in the PI_UserStages2.dat stage database and in the main window of PIMikroMove.

## 10.3 Parameter Overview

***INFORMATION***

The password for saving the parameter values in the non-volatile memory reads 100.

| **Parameter ID (hexa-decimal)** | **Data type** | **Parameter name** | **Description** |
|---|---|---|---|
| 0x1 | INT | P-Term | 0 to 32767; details see "Servo Algorithm and Other Control Value Corrections" (p. 27). |
| 0x2 | INT | I-Term | 0 to 32767; details see "Servo Algorithm and Other Control Value Corrections" (p. 27). |
| 0x3 | INT | D-Term | 0 to 32767; details see "Servo Algorithm and Other Control Value Corrections" (p. 27). |
| 0x4 | INT | I-Limit | 0 to 32767; details see "Servo Algorithm and Other Control Value Corrections" (p. 27). |
| 0x5 | INT | Kvff | 0 to 32767; details see "Servo Algorithm and Other Control Value Corrections" (p. 27). |
| 0x8 | FLOAT | Maximum Position Error (Phys. Unit) | Maximum position error<br>Is used for the detection of motion errors. Details see "Motion Error" (p. 93). |
| 0x9 | INT | Maximum Motor Output | Maximum control value for driving the axis (dimensionless)<br>0 to 32767 |
| 0xA | FLOAT | Maximum Closed-Loop Velocity (Phys. Unit/s) | Maximum velocity in closed-loop operation<br>Specifies the maximum value for parameter 0x49. |
| 0xB | FLOAT | Closed-Loop Acceleration (Phys. Unit/s$^2$) | Acceleration in closed-loop operation<br>Is limited by parameter 0x4A.<br>Details see "Generation of Dynamics Profile" (p. 24). |
| 0xC | FLOAT | Closed-Loop Deceleration (Phys. Unit/s$^2$) | Deceleration in closed-loop operation<br>Is limited by parameter 0x4B.<br>Details see "Generation of Dynamics Profile" (p. 24). |

| Parameter ID (hexa-decimal) | Data type | Parameter name | Description |
|---|---|---|---|
| 0xE | INT | Numerator Of The Counts-Per-Physical-Unit Factor | Numerator and denominator of the factor for counts per physical length unit<br>Details see "Physical Units" (p. 22). |
| 0xF | INT | Denominator Of The Counts-Per-Physical-Unit Factor | |
| 0x13 | INT | Is Rotary Stage? | Is this a rotary stage?<br>0 = Not a rotary stage<br>1 = Rotary stage<br>No evaluation by the C-863, but only by the PC software: PIMikroMove determines on the basis of this value which motions are permissible. |
| 0x14 | INT | Has Reference? | Does the stage have a reference point switch?<br>Details see "Reference Point Switch Detection" (p. 31). |
| 0x15 | FLOAT | Maximum Travel In Positive Direction (Phys. Unit) | Soft limit in positive direction<br>See examples in "Travel Range and Soft Limits" (p. 35). |
| 0x16 | FLOAT | Value At Reference Position (Phys. Unit) | Position value on the reference point switch<br>See examples in "Travel Range and Soft Limits" (p. 35). |
| 0x17 | FLOAT | Distance From Negative Limit To Reference Position (Phys. Unit) | Distance between reference point switch and negative limit switch<br>See examples in "Travel Range and Soft Limits" (p. 35). |
| 0x18 | INT | Limit Mode | Signal logic of the limit switches<br>Details see "Limit Switch Detection" (p. 32). |
| 0x1A | INT | Has Brake? | Does the stage have a brake?<br>0 = No brake present<br>1 = Brake present. In this case, the switching on/off the servo mode and activating/deactivating the brake are intercoupled, see `BRA` (p. 161) and `SVO` (p. 233). |

| Parameter ID (hexa-decimal) | Data type | Parameter name | Description |
|---|---|---|---|
| 0x2F | FLOAT | Distance From Reference Position To Positive Limit (Phys. Unit) | Distance between reference point switch and positive limit switch<br>See examples in "Travel Range and Soft Limits" (p. 35). |
| 0x30 | FLOAT | Maximum Travel In Negative Direction (Phys. Unit) | Soft limit in negative direction<br>See examples in "Travel Range and Soft Limits" (p. 35). |
| 0x31 | INT | Invert Reference? | Should the reference signal be inverted?<br>Details see "Reference Point Switch Detection" (p. 31). |
| 0x32 | INT | Has No Limit Switches? | Does the stage have limit switches?<br>Details see "Limit Switch Detection" (p. 32). |
| 0x33 | INT | Motor Offset Positive | Offset for the positive direction of motion<br>Details see "Servo Algorithm and Other Control Value Corrections" (p. 27). |
| 0x34 | INT | Motor Offset Negative | Offset for the negative direction of motion<br>Details see "Servo Algorithm and Other Control Value Corrections" (p. 27). |
| 0x36 | INT | Settling Window (encoder counts) | Settling window around the target position<br>Details see "On-Target State" (p. 30). |
| 0x3C | CHAR | Stage Name | Stage name<br>Maximum 20 characters; default value: DEFAULT_STAGE |
| 0x3F | FLOAT | Settling Time (s) | Delay time for setting the on-target state.<br>Details see "On-Target State" (p. 30). |
| 0x47 | INT | Reference Travel Direction | Default direction for the reference move<br>Details see "Reference Point Definition" (p. 38). |
| 0x48 | INT | Motor Drive Offset | Velocity-dependent offset<br>Details see "Servo Algorithm and Other Control Value Corrections" (p. 27). |

| Parameter ID (hexa-decimal) | Data type | Parameter name | Description |
|---|---|---|---|
| 0x49 | FLOAT | Closed-Loop Velocity (Phys. Unit/s) | Velocity in closed-loop operation<br>Is limited by parameter 0xA<br>Details see "Generation of Dynamics Profile" (p. 24). |
| 0x4A | FLOAT | Maximum Closed-Loop Acceleration (Phys. Unit/s$^2$) | Maximum acceleration in closed-loop operation<br>Specifies the maximum value for parameter 0xB. |
| 0x4B | FLOAT | Maximum Closed-Loop Deceleration (Phys. Unit/s$^2$) | Maximum deceleration in closed-loop operation<br>Specifies the maximum value for parameter 0xC. |
| 0x4D | INT | Servo Window Mode | Present for reason of compatibility only. |
| 0x50 | FLOAT | Velocity For Reference Moves (Phys. Unit/s) | Velocity for reference move<br>Details see "Reference Point Definition" (p. 38). |
| 0x5A | INT | Numerator Of The Servo-Loop Input Factor | Numerator and denominator of the servo loop input factor<br>Details see "Servo Algorithm and Other Control Value Corrections" (p. 27). |
| 0x5B | INT | Denominator Of The Servo-Loop Input Factor | |
| 0x5C | INT | Source Of Reference Signal | Reference signal source for the `FRF` or `FED` commands<br>Details see "Commands and Parameters for Digital Inputs" (p. 106) and "Using Digital Input Signals as Switch Signals" (p. 109). |
| 0x5D | INT | Source Of Negative Limit Signal | Reference signal source for the `FNL` or `FED` commands<br>Details see "Commands and Parameters for Digital Inputs" (p. 106) and "Using Digital Input Signals as Switch Signals" (p. 109). |
| 0x5E | INT | Source Of Positive Limit Signal | Reference signal source for the `FPL` or `FED` commands<br>Details see "Commands and Parameters for Digital Inputs" (p. 106) and "Using Digital Input Signals as Switch Signals" (p. 109). |

| Parameter ID (hexa-decimal) | Data type | Parameter name | Description |
|---|---|---|---|
| 0x5F | INT | Invert Digital Input Used For Negative Limit | Inverts the polarity of the digital inputs that are used for the source of the negative limit switch signal.<br>Details see "Commands and Parameters for Digital Inputs" (p. 106) and "Using Digital Input Signals as Switch Signals" (p. 109). |
| 0x60 | INT | Invert Digital Input Used For Positive Limit | Inverts the polarity of the digital inputs that are used for the source of the positive limit switch signal.<br>Details see "Commands and Parameters for Digital Inputs" (p. 106) and "Using Digital Input Signals as Switch Signals" (p. 109). |
| 0x61 | INT | Invert Direction Of Motion For Joystick-Controlled Axis? | Shall the direction of motion for joystick-controlled axes be inverted?<br>Details see "Commands and Parameters for Joystick Control" (p. 113). |
| 0x63 | FLOAT | Distance Between Limit And Hard Stop (Phys. Unit) | Distance between internal limit switch and hard stop<br>Details see "Reference Point Definition" (p. 38). |
| 0x70 | INT | Reference Signal Type | Reference signal type<br>Details see "Reference Point Switch Detection" (p. 31). |
| 0x71 | INT | D-Term Delay (No. Of Servo Cycles) | D-Term delay<br>Details see "Servo Algorithm and Other Control Value Corrections" (p. 27). |
| 0x72 | INT | Ignore Macro Error? | Ignore macro error?<br>Details see "Commands and Parameters for Macros" (p. 122). |
| 0x94 | FLOAT | Notch Filter Frequency 1 (Hz) | Frequency of the first notch filter<br>Details see "Servo Algorithm and Other Control Value Corrections" (p. 27). |
| 0x95 | FLOAT | Notch Filter Edge 1 | Rise of the edge of the first notch filter<br>Details see "Servo Algorithm and Other Control Value Corrections" (p. 27). |

| Parameter ID (hexa-decimal) | Data type | Parameter name | Description |
|---|---|---|---|
| 0x400 | INT | Number of Servo Parameter Sets | Present for reason of compatibility only. |
| 0x401 | INT | P-Term 0 | Parameter set 0: Present for reason of compatibility only. |
| 0x402 | INT | I-Term 0 | |
| 0x403 | INT | D-Term 0 | |
| 0x404 | INT | I-Limit 0 | |
| 0x405 | INT | Kvff 0 | |
| 0x406 | INT | Window Enter 0 (encoder counts) | Present for reason of compatibility only.<br>Value identical to the value of parameter 0x36. |
| 0x407 | INT | Window Exit 0 (encoder counts) | Present for reason of compatibility only. |
| 0x411 | INT | P-Term 1 | Parameter set 1: Present for reason of compatibility only.<br>Values identical to the values of parameters 0x1 to 0x5. |
| 0x412 | INT | I-Term 1 | |
| 0x413 | INT | D-Term 1 | |
| 0x414 | INT | I-Limit 1 | |
| 0x415 | INT | Kvff 1 | |
| 0x416 | INT | Window Enter 1 (encoder counts) | Present for reason of compatibility only. |
| 0x417 | INT | Window Exit 1 (encoder counts) | |
| 0x421 | INT | P-Term 2 | Parameter set 2: Present for reason of compatibility only. |
| 0x422 | INT | I-Term 2 | |
| 0x423 | INT | D-Term 2 | |
| 0x424 | INT | I-Limit 2 | |
| 0x425 | INT | Kvff 2 | |
| 0x426 | INT | Window Enter 2 (encoder counts) | |
| 0x427 | INT | Window Exit 2 (encoder counts) | |

| **Parameter ID (hexa-decimal)** | **Data type** | **Parameter name** | **Description** |
|---|---|---|---|
| 0x431 | INT | P-Term 3 | Parameter set 3: Present for reason of compatibility only. |
| 0x432 | INT | I-Term 3 | |
| 0x433 | INT | D-Term 3 | |
| 0x434 | INT | I-Limit 3 | |
| 0x435 | INT | Kvff 3 | |
| 0x436 | INT | Window Enter 3 (encoder counts) | |
| 0x437 | INT | Window Exit 3 (encoder counts) | |
| 0x441 | INT | P-Term 4 | Parameter set 4: Present for reason of compatibility only. |
| 0x442 | INT | I-Term 4 | |
| 0x443 | INT | D-Term 4 | |
| 0x444 | INT | I-Limit 4 | |
| 0x445 | INT | Kvff 4 | |
| 0x446 | INT | Window Enter 4 (encoder counts) | |
| 0x447 | INT | Window Exit 4 (encoder counts) | |
| 0x07000000 | FLOAT | Range Limit Min | Additional soft limit for the negative direction of motion (physical unit)<br>Details see "Travel Range and Soft Limits" (p. 33). |
| 0x07000001 | FLOAT | Range Limit Max | Additional soft limit for the positive direction of motion (physical unit)<br>Details see "Travel Range and Soft Limits" (p. 33). |
| 0x07000601 | CHAR | Axis Unit | Unit symbol<br>Details see "Physical Units" (p. 22). |

# 11 Maintenance

## In this Chapter

## 11.1 Cleaning the C-863

***NOTICE***

**Short circuits or flashovers!**

The C-863 contains electrostatic sensitive devices that can be damaged by short circuits or flashovers when cleaning fluids enter the case.

- Before cleaning, remove the C-863 from the power source by pulling the power plug.
- Prevent cleaning fluid from entering the case.

- When necessary, clean the surfaces of the C-863 case with a towel that has been lightly dampened with a mild cleanser or disinfectant.
- Do **not** use any organic solvents.

## 11.2 Updating Firmware

***INFORMATION***

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

Example of a C-863 response:

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

- C-863.11: Device name
- 1.2.0.0: Firmware version

▪

***INFORMATION***

If the C-863 is in firmware update mode (DIP switch 8 in "ON" (upper) position), the DIP switch settings for baud rate and controller address are ignored. When communication is established with the ***TMS320F28xx Updater*** program, the baud rate is set automatically.

***INFORMATION***

When updating the firmware, the parameter settings of the C-863 are not changed. When new parameters are introduced with the firmware update, a special initialization for the values of the new parameters is required prior to operating the C-863.

***INFORMATION***

When updating the firmware the macros saved on the C-863 are retained.

### Prerequisite

✓ You have connected the C-863 to the PC via the RS-232 interface (p. 60).

✓ You have made sure that the C-863 is **not** a part of a daisy chain network.

✓ You have made sure that **no** cable is connected to the **RS-232 Out** socket.

✓ You have read and understood the documentation that you received from our customer service department together with the current flash and bootloader files. You have learned from the documentation whether new parameters are introduced with the firmware update.

## Tools and accessories

- Current flash file (contains the DSP firmware); available on request from our customer service department (p. 291)
- Current bootloader file; available on request from our customer service department (p. 291)
- Product CD with the ***TMS320F28xx Updater*** program, included in the scope of delivery

## Installing TMS320F28xx Updater and firmware files on the PC

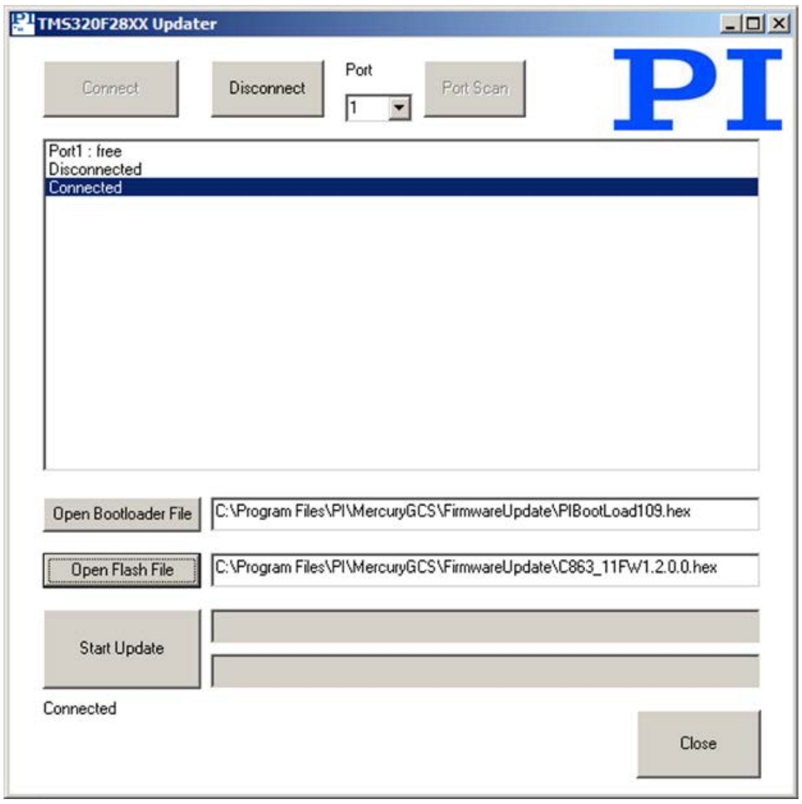
1. Insert the product CD in the drive of the PC.
2. If the installation wizard does not start automatically:

   Start the installation wizard by clicking on the icon in the root directory of the CD.
3. Follow the instructions on the screen.
   a) Depending on availability, select ***Custom*** for the user-defined installation or ***Modify*** for the completion of an installation already present on the PC.
   b) Select ***Firmware Updater*** for the ***TMS320F28xx Updater*** program.
4. Continue to follow the instructions on the screen to finish the installation.
5. Copy the current flash file and the current bootloader file to the installation directory of the ***TMS320F28xx Updater*** program on the PC.

## Updating firmware of the C-863

1. If new parameters are introduced with the firmware update: Save the current parameter values of the C-863 in a text file (p. 265).
2. Switch off the C-863 by pulling the power cord of the power supply.
3. Set DIP switch 8 on the C-863 to the firmware update mode (ON position) (p. 71).
4. Switch on the C-863 by connecting the power cord of the power supply to the power socket.
5. Start the ***TMS320F28xx Updater*** program on the PC.
6. Click on the ***Connect*** button.

Communication between C-863 and PC is established. The message `Connected` appears in the message list of the program window.



```
Port1 : free
Disconnected
Connected
```

C:\Program Files\PI\MercuryGCS\FirmwareUpdate\PIBootLoad109.hex

C:\Program Files\PI\MercuryGCS\FirmwareUpdate\C863_11FW1.2.0.0.hex

Connected

If communication cannot be established:

a) Click on the ***Port Scan*** button.
b) If necessary, change the setting in the ***Port*** selection field.
c) Click on ***Connect*** again.

7. Select the current bootloader file:

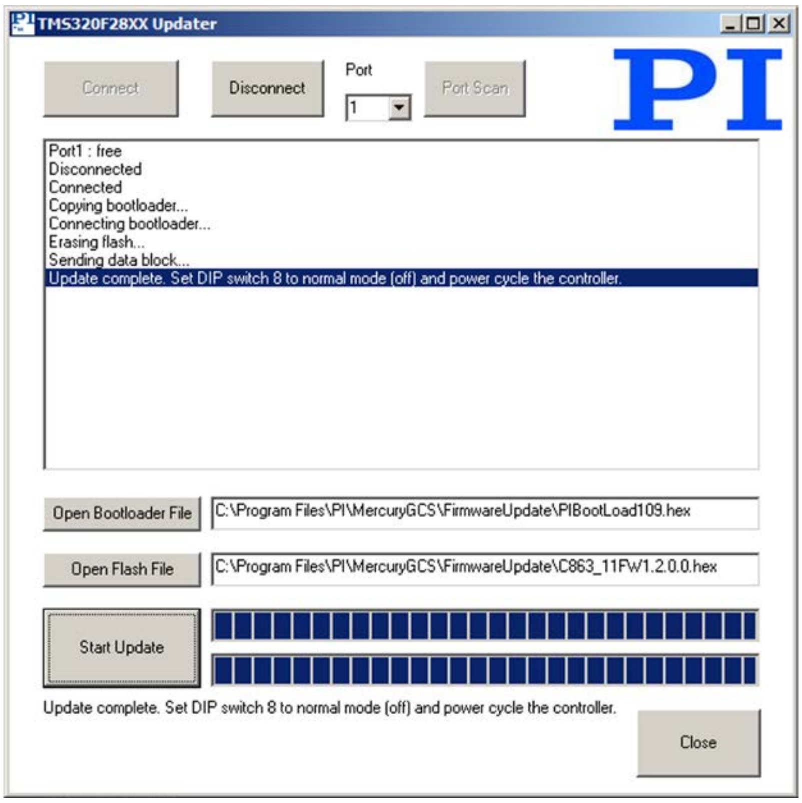
   a) Click on the ***Open Bootloader File*** button.
   b) In the file selection window, switch to the installation directory of the ***TMS320F28xx Updater*** program.
   c) Here double click on the current bootloader file.

8. Select the current flash file:

   a) Click on the ***Open Flash File*** button.
   b) In the file selection window, switch to the installation directory of the ***TMS320F28xx Updater*** program.
   c) Here double click on the current flash file (C863_11FWz.z.z.z.hex, where *z* represents the firmware version).

9. Start the firmware update by clicking on the ***Start Update*** button.
   – If a message box opens, follow the instructions given there.

   The firmware of the C-863 is updated. The progress of the update is displayed in the message list and by the two progress bars. The update was successful if the message `Update complete. Set DIP switch 8 to normal mode (off) and power cycle the controller.` appears in the message list.



```
Port1 : free
Disconnected
Connected
Copying bootloader...
Connecting bootloader...
Erasing flash...
Sending data block...
Update complete. Set DIP switch 8 to normal mode (off) and power cycle the controller.
```

C:\Program Files\PI\MercuryGCS\FirmwareUpdate\PIBootLoad109.hex

C:\Program Files\PI\MercuryGCS\FirmwareUpdate\C863_11FW1.2.0.0.hex

Update complete. Set DIP switch 8 to normal mode (off) and power cycle the controller.

10. Close the ***TMS320F28xx Updater*** program by clicking on the ***Close*** button.
11. Switch off the C-863 by pulling the power cord of the power supply.
12. Set DIP switch 8 on the C-863 to normal operation (OFF position) (p. 71).
13. Switch on the C-863 by connecting the power cord of the power supply to the power socket.

Were new parameters introduced with firmware update?

- If no: Firmware update is finished.
- If yes: A special initialization for the values of the new parameters is required prior to operating the C-863, see below.

## Initializing new parameters of the C-863

1. Start PITerminal or PIMikroMove on the PC and, if necessary, open the window for sending commands.
2. Make sure that the current parameter values of the C-863 have been saved in a text file (see "Saving Parameter Values in a Text File" (p. 265)).

   When initializing the new parameters, **all** parameters of the C-863 are set to their default settings. Parameter values that are not saved are lost during the initialization process as a result.
3. Initialize the new parameters by sending the `ZZZ 100 parameter` command.
4. Adapt the parameter values (see General Procedure for Changing Parameters (p. 266)):
   - 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.

# 12 Troubleshooting

| Problem | Possible Causes | Solution |
|---|---|---|
| The stage does not move | The cable is not connected correctly | ➢ Check the cable connections. |
| | The stage or the stage cable is defective | ➢ If available, replace the defective stage with another stage and test the new combination. |
| | Stage not connected to power supply | Stages with integrated PWM amplifier are supplied via a separate power supply.<br>➢ If the stage has an integrated PWM amplifier, connect it to a suitable power supply.<br>➢ Make sure that the power supply is functioning properly. |
| | Limit switch signal logic set incorrectly | In order for the stage to be able to move, all settings of the C-863 must correspond to the limit switch logic level of the stage; see "Limit Switch Detection" (p. 32).<br>➢ Set DIP switch 7 appropriately; see "Logic Level of the Limit Switches" (p. 70).<br>➢ Set the ***Limit Mode*** parameter (ID 0x18) appropriately; see "Changing Parameters: General Procedure (p. 266)". |
| | Limit switch signals not compatible with the C-863 | Stages from third-party suppliers may use unsuitable limit switch signals.<br>➢ Contact the customer service department (p. 291) and the manufacturer of the stage. |
| | Incorrect configuration | ➢ Check the parameter settings of the C-863 with the `SPA?` (volatile memory) and `SEP?` (non-volatile memory) commands. Details about parameter settings see "Adapting settings" (p. 263). |
| | Incorrect command or incorrect syntax | ➢ Send the `ERR?` command and check the error code this returns. |

| Problem | Possible Causes | Solution |
|---|---|---|
| | Incorrect axis commanded | An axis identifier is even required in commands on systems with only one axis.<br>➢ Make sure that the correct axis identifier is used and that the commanded axis belongs to the correct stage. |
| | Joystick control is enabled | Motion commands are not allowed when a joystick is enabled for the axis.<br>➢ Disable the joystick with the `JON` command (p. 201). |
| Stage executes unintentional motion | Joystick is not connected, but enabled in the C-863 | ➢ Enable the joystick in the software only if a joystick is actually connected to the C-863 (p. 114). |
| | Joystick not calibrated | ➢ Calibrate the joystick (p. 116). |
| | Start-up macro is executed | ➢ Check whether a macro is specified as a start-up macro and, if necessary, cancel the selection of the start-up macro (p. 123). |
| | Stage brake deactivated with the `BRA` command (p. 161) while servo mode switched off | ➢ Secure the stage against unintentional motions before you deactivate the brake by command! |
| Stage is oscillating or positioned inaccurately | The load was changed. | ➢ Readjust the system according to the changed load (p. 90). |

| Problem | Possible Causes | Solution |
|---|---|---|
| Stage is oscillating already during the reference move | Very high load on the stage | In case of a very high load, proceed with PIMikroMove during the reference move as follows:<br>1. Do **not** start the reference move in the ***Start up axes*** step, but click on ***Close*** to close the ***Start up controller*** window instead.<br>2. In the main window, open the single axis window for the stage connected by selecting the stage in the ***View > Single Axis Window*** menu.<br>3. Expand the view of the single axis window by clicking on the **>** button at the right edge of the window.<br>4. With the ***Servo*** check box, make sure that the servo mode is switched on.<br>5. Start the reference move by clicking on one of the ***Reference…*** buttons.<br>6. If the stage is oscillating: Immediately stop the reference move in the ***Reference Axes*** dialog, close the dialog and switch off the servo mode by removing the check from the respective check box in the single axis window.<br>7. Enter new values for the servo-control parameter, see "Optimizing the Servo-Control Parameters" (p. 90).<br>8. Re-start the reference move.<br>9. If the stage is still oscillating, repeat steps 6 to 8 until the reference move is finished successfully without oscillations. |
| Communication with the controller fails | The wrong communication cable is used or it is defective | ➢ Use a null-modem cable for the RS-232 connection (p. 60).<br>➢ If necessary, check whether the cable works on a fault-free system. |

| Problem | Possible Causes | Solution |
|---|---|---|
| | USB driver not installed | In order to be able to establish communication between C-863 and PC via USB interface, drivers from the product CD have to be installed. If communication via USB cannot be established or the PC operating system reports that new hardware was found:<br>1. Log in on the PC with administrator rights.<br>2. Insert the product CD in the drive of the PC.<br>3. Follow the instructions on the PC screen or open the ***Properties of PI C-863 controllers*** window in the appropriate manner.<br>4. Select the drivers in the \USB_Driver directory on the product CD when prompted to do so.<br>If the C-863 is still not detected by the PC after successfully installing the drivers:<br>➢ Pull the USB cable off of the C-863 and plug it back in. |
| | Baud rate not configured correctly | ➢ Check the settings of DIP switches 5 and 6 for the baud rate (p. 70).<br>➢ In a daisy chain network make sure that the same baud rate is set for every controller. |
| | Controller address not configured correctly | ➢ Check the settings of DIP switches 1 to 4 for the controller address (p. 69). |
| | A different program is accessing the interface. | ➢ Close the other program. |
| | Problems with special software | ➢ Check whether the system works with different software, such as a terminal program or a development environment.<br>You can test the communication by starting a terminal program (such as e. g. PI Terminal) and entering `*IDN?` or `HLP?`.<br>➢ Make sure that you end the command with an LF (line feed).<br>A command is not executed until the LF has been received. |

| Problem | Possible Causes | Solution |
|---|---|---|
| The customer software does not run with the PI drivers | Incorrect combination of driver routines/Vis | ➢ Check whether the system works with a terminal program.<br>If so:<br>➢ Read the information in the corresponding software manual and compare the sample code on the product CD with your program code. |
| LEDs do not light up even though the C-863 is switched on | Firmware update mode set | 1. Switch off the C-863 by pulling the power cord of the power supply.<br>2. Set DIP switch 8 on the C-863 to normal operation (p. 71) (OFF) position.<br>3. Switch on the C-863 by connecting the power cord of the power supply to the power socket. |

If the problem that occurred with your system is not listed in the table above or it cannot be solved as described, contact our customer service department (p. 291).

# 13 Customer Service

For inquiries and orders, contact your PI sales engineer or send us an e-mail (mailto:info@pi.ws).

If you have questions concerning your system, have the following information ready:

- Product codes and serial numbers of all products in the system
- Firmware version of the controller (if present)
- Version of the driver or the software (if present)
- Operating system on the PC (if present)

The latest versions of the user manuals are available for downloading (p. 5) on our website.

# 14 Technical Data

## In this Chapter

## 14.1 Specifications

### 14.1.1 Data Table

| | **C-863.11** |
|---|---|
| Function | DC servo-motor controller, 1 channel |
| Channels | 1 |
| **Motion and control** | |
| Servo characteristics | PID controller, parameter changes on the fly |
| Servo cycle time | 50 µs |
| Profile generator | Trapezoid velocity profile |
| Encoder input | AB (quadrature) single-ended or differential TTL signal acc. to RS-422; 60 MHz |
| Stall detection | Servo off, triggered by programmable position error |
| Limit switches | 2 × TTL (polarity programmable) |
| Reference point switch | 1 × TTL |
| Motor brake | 1 × TTL, software controlled |
| **Electrical properties** | |
| Max. output voltage* | 0 to ±15 V for direct control of DC motor |
| Max. output power | 30 W |
| Current limitation | 2 A |

| | |
|---|---|
| **Interface and operation** | |
| Communication interfaces | USB; RS-232, Sub-D 9-pin (m) |
| Motor connector | Sub-D 15-pin (f) |
| Controller network | Up to 16 units** on a single interface |
| I/O ports | 4 analog/digital in, 4 digital out (TTL), 5 V TTL |
| Command set | PI General Command Set (GCS) |
| User software | PIMikroMove |
| Software drivers | LabVIEW driver, shared libraries for Windows and Linux |
| Supported functionality | Point-to-point motion, start-up macro, data recorder for recording parameters as motor input voltage, velocity, position or position error; internal safety circuitry: watchdog timer |
| Manual control | Optional: Pushbutton box, joystick (for 2 axes), Y-cable for 2-D motion |
| **Miscellaneous** | |
| Operating voltage | 15 to 30 V, in the scope of delivery: external power supply 15 V / 2 A |
| Max. operating current | 80 mA plus motor current (max. 3 A) |
| Operating temperature range | 5 to 50°C |
| Ground | 0.3 kg |
| Dimensions | 130 mm × 76 mm × 40 mm |

* The output voltage depends on the connected power supply.

** 16 units via USB; 6 units via RS-232.

### 14.1.2 Maximum Ratings

The C-863 is designed for the following maximum ratings:

| **Input on:** | **Maximum Operating Voltage** ⚠ | **Operating Frequency** ⚠ | **Maximum Current Consumption** ⚠ |
|---|---|---|---|
| Barrel connector socket | 30 V | ⎓ | 3 A |

| **Output on:** | **Maximum Output Voltage** ⚠ | **Maximum Output Current** ⚠ | **Maximum Output Frequency** ⚠ |
|---|---|---|---|
| Sub-D 15-pin (f): pins 2 and 9 | = operating voltage | 2.5 A | 36 kHz (PWM) |
| Sub-D 15-pin (f): pins 3 and 11 | 5 V TTL | 10 mA | 36 kHz (PWM) |

### 14.1.3 Ambient Conditions and Classifications

The following ambient conditions and classifications must be observed for the C-863:

| | |
|---|---|
| Area of application | For indoor use only |
| Maximum altitude | 2000 m |
| Relative humidity | Highest relative humidity 80% for temperatures up to 31°C<br>Decreasing linearly to 50% relative humidity at 40°C |
| Storage temperature | 0°C to 70°C |
| Transport temperature | –25°C to +85°C |
| Overvoltage category | II |
| Protection class | I |
| Degree of pollution | 2 |
| Measurement category | I |
| Degree of protection according to IEC 60529 | IP20 |

## 14.2 System Requirements

The following system requirements must be met to operate the C-863:

- PC with Windows (XP, Vista, 7) or Linux operating system
- Communication interface to the PC:
  - Free COM port on the PC

  - or -

  - USB A socket on the PC
- C-863 with power supply
- Mechanical system (stage) with DC motor
- RS-232 null-modem cable or USB cable for connecting the controller to the PC
- Product CD with PC software

## 14.3 Dimensions

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

Standard tolerance according to DIN ISO 2768 - f - H

Roughness Ra 1.6

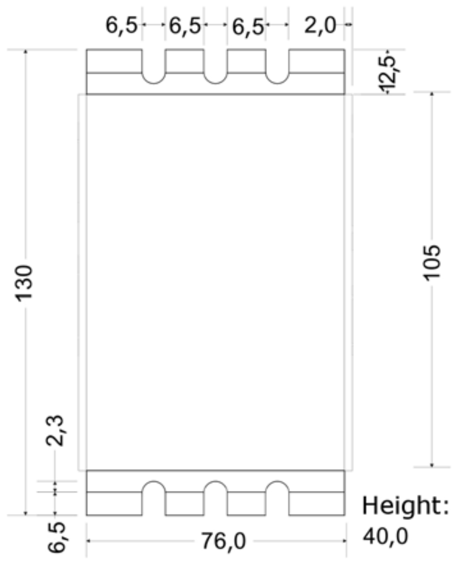
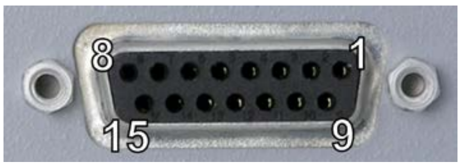


*Figure 35: C-863, dimensions in millimeters*

## 14.4 Pin Assignment

### 14.4.1 DC Motor only

**Sub-D socket, 15-pin, female**



| Pin | Function |
|---|---|
| 1 | Output: Programmable motor brake (0 or + 5 V) |
| 2 | Output: Motor + (differential; power PWM); for stages without PWM amplifier |
| 3 | Output: PWM magnitude (TTL); for stages with PWM amplifier |
| 4 | Output: +5 V |
| 5 | Input: Positive limit switch |
| 6 | GND limit switch |
| 7 | Input: Encoder: A ( - ) |
| 8 | Input: Encoder: B ( - ) |
| 9 | Output: Motor – (differential; power PWM); for stages without PWM amplifier |
| 10 | Power GND |
| 11 | Output: PWM sign (TTL); for stages with PWM amplifier |
| 12 | Input: Negative limit switch |
| 13 | Input: Reference point switch |
| 14 | Input: encoder: A (+) / ENCA |
| 15 | Input: encoder: B (+) / ENCB |

## 14.4.2 I/O

### Mini-DIN socket, 9-pin, female

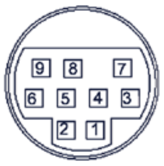


*Figure 36: I/O socket, Mini-DIN, 9-pin*

| 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) |

## 14.4.3 C-170.IO Cable for Connecting to the I/O Socket

### Mini-DIN connector, 9-pin, male, open end

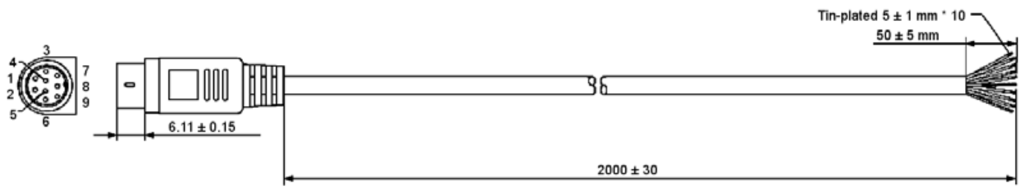


*Figure 37: C-170.IO cable*

### Specifications

Temperature range: -25 °C to +85 °C
Nominal current: 1 A AC/DC
Insulation resistance: 50 MW min.
Nominal voltage: 50 V AC/DC
Voltage impulse: 500 V AC for 1 minute

| Pin | Wire color | Function on the I/O socket of the C-863 |
|---|---|---|
| 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 |

### 14.4.4 Joystick

#### Mini-DIN socket, 6-pin, female (PS/2)

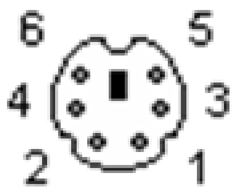


*Figure 38: Front view of Mini-DIN socket*

| Pin | Function |
|---|---|
| 1 | GND |
| 2 | Input: axis 1 of joystick device 2 (-10 to 10 V) |
| 3 | Output: Vcc (3.3 V) |
| 4 | Input: axis 1 of joystick device 1 (0 to 3.3 V) |
| 5 | Not connected |
| 6 | Input: joystick button 1 of joystick device 1 (0 or 3.3 V) |

## 14.4.5 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.

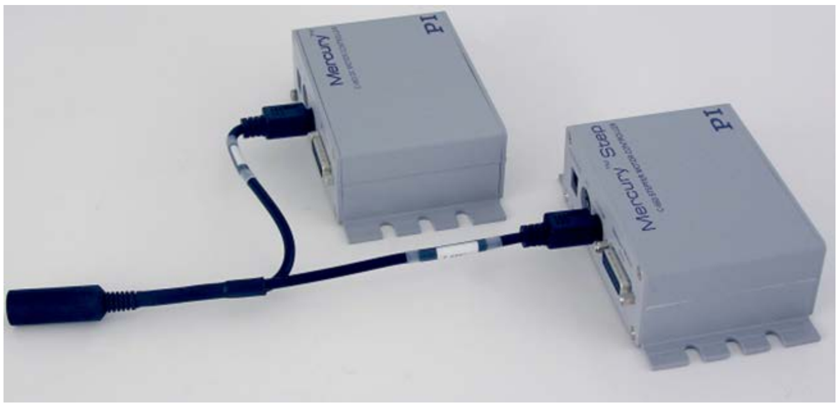


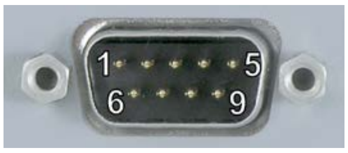
*Figure 39: Y cable C-819.20Y for joystick with 2 controllers*

### Mini-DIN connector, 6-pin, female on 2 Mini-DIN connectors, 6-pin, male

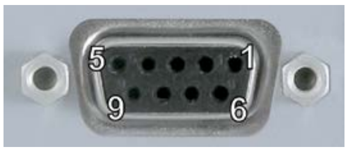
| **Mini-DIN 6-pin, female (to joystick)** | **Signal** | **Mini-DIN, 6-pin, male, X branch (to controller 1)** | **Mini-DIN, 6-pin, male, Y branch (to controller 2)** |
|---|---|---|---|
| Pin 1 | GND | Pin 1 | Pin 1 |
| Pin 2 | Button for joystick Y axis | Not connected | Pin 6 |
| Pin 3 | Joystick 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 |

### 14.4.6 RS-232 In and RS-232 Out

**RS-232 In: Sub-D panel plug, 9-pin, male**



**RS-232 Out: Sub-D socket, 9-pin, female**



| Pin | Function |
|---|---|
| 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 together in the C-863 1:1.

***INFORMATION***

In a daisy chain network that is 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-863 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-863 copies the signal of the TxD line via USB to the PC.

### 14.4.7 Power Supply Connection 15-30 VDC

#### Barrel connector socket

| Pin | Function |
|---|---|
| Center pin | Input: 15 to 30 VDC |
| Outer conductor | GND (power) |

# 15 Old Equipment Disposal

In accordance with the applicable EU law, electrical and electronic equipment may not be disposed of with unsorted municipal wastes in the member states of the EU.

When disposing of your old equipment, observe the international, national and local rules and regulations.

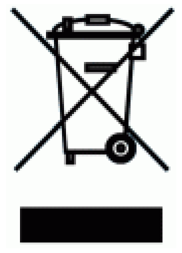
To meet the manufacturer’s product responsibility with regard to this product, Physik Instrumente (PI) GmbH & Co. KG ensures environmentally correct disposal of old PI equipment that was first put into circulation after 13 August 2005, free of charge.

If you have old PI equipment, you can send it postage-free to the following address:

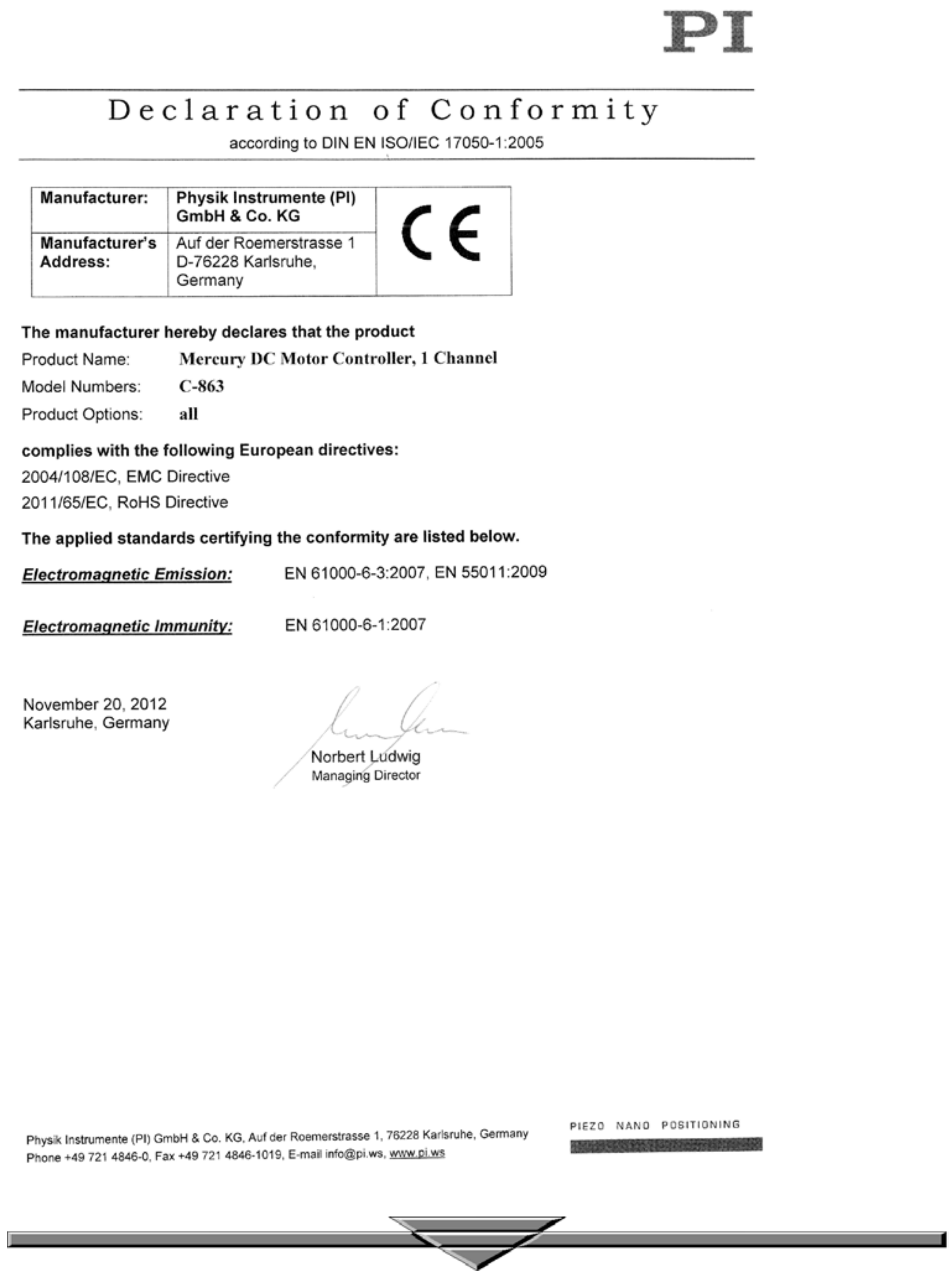
Physik Instrumente (PI) GmbH & Co. KG

Auf der Römerstr. 1

D-76228 Karlsruhe, Germany



# 16 EC Declaration of Conformity


PI

Declaration of Conformity

according to DIN EN ISO/IEC 17050-1:2005

| | | |
|---|---|---|
| **Manufacturer:** | **Physik Instrumente (PI) GmbH & Co. KG** | CE |
| **Manufacturer's Address:** | Auf der Roemerstrasse 1 D-76228 Karlsruhe, Germany | |

**The manufacturer hereby declares that the product**

| | |
|---|---|
| Product Name: | **Mercury DC Motor Controller, 1 Channel** |
| Model Numbers: | **C-863** |
| Product Options: | **all** |

**complies with the following European directives:**

2004/108/EC, EMC Directive

2011/65/EC, RoHS Directive

**The applied standards certifying the conformity are listed below.**

| | |
|---|---|
| ***Electromagnetic Emission:*** | EN 61000-6-3:2007, EN 55011:2009 |
| ***Electromagnetic Immunity:*** | EN 61000-6-1:2007 |

November 20, 2012
Karlsruhe, Germany

[illegible]

Norbert Ludwig
Managing Director

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

PIEZO NANO POSITIONING