MS208E User Manual

C-663 Mercury Step

Stepper Motor Controller
Release: 1.0.0 Date: 09.05.2011

This document describes the following product:

- C-663.11
  Mercury Step Stepper Motor Controller, 1 Channel, with Wide-Range Power Supply (24 V)

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

PI

Declaration of Conformity

according to DIN EN ISO/IEC 17050-1

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 Step Stepper Motor Controller, 1 Channel**

Model Numbers: **C-663**

Product Options: **all**

complies with the following European directives:

2006/95/EC, Low Voltage Directive (LVD)

2004/108/EC, EMC Directive

The applied standards certifying the conformity are listed below.

Electromagnetic Emission: EN 61000-6-3, EN 55011 (05/98)+ A1 (08/99) class A, EN 55022 (09/98)

Electromagnetic Immunity: EN 61000-6-1

Safety (Low Voltage Directive): EN 61010-1

3 May 2011
Karlsruhe, Germany

Dr. Karl Spanner
President

Physik Instrumente (PI) GmbH & Co. KG is the owner of the following company names and trademarks:
PI®, PIC®, PICMA®, PILine®,PIFOC®, PiezoWalk®, NEXACT®, NEXLINE®, NanoCube®, NanoAutomation®

The following designations are protected company names or registered trademarks of third parties:
Microsoft, Windows, LabView

The products described in this document are in part protected by the following patents:
US Pat. No. 6,765,335
German Patent No. 10154526

© 2011 Physik Instrumente (PI) GmbH & Co. KG, Karlsruhe, Germany. The text, photographs and drawings in this manual are proteced by copyright. With regard thereto, Physik Instrumente (PI) GmbH & Co. KG reserves all rights. Use of said text, photographs and drawings is permitted only in part and only upon citation of the source.

First printing: 09.05.2011
Document number: MS208E, CBo, Release 1.0.0

Subject to change without notice. This manual is superseded by any new release. The latest release is available for download on our website (www.pi.ws).

About this Document

Users of this Manual

This manual is designed to help the reader to operate the C-663 stepper motor controller. It assumes that the reader has a fundamental understanding of basic stepper motor systems, as well as motion control concepts and applicable safety procedures.
The manual describes the physical specifications and dimensions of the C-663 as well as the software and hardware installation procedures and the commands which are required to put the associated motion system into operation.

Conventions

The notes and symbols used in this manual have the following meanings:

DANGER

Indicates the presence of high voltage (> 50 V). Calls attention to a procedure, practice or condition which, if not correctly performed or adhered to, could result in injury or death.

WARNING

Calls attention to a procedure, practice or condition which, if not correctly performed or adhered to, could result in injury or death.

CAUTION

Calls attention to a procedure, practice, or condition which, if not correctly performed or adhered to, could result in damage to equipment.

NOTE

Provides additional information or application hints.

The software tools and the mechanical systems which might be mentioned in this document are described in their own manuals. All documents are available as PDF files. Updated releases are available for download at www.pi.ws or via e-mail: contact your Physik Instrumente Sales Engineer or write to info@pi.ws.

Related Documents

Mercury_GCS_LabVIEW_MS206E	LabView VIs based on PI GCS command set
PIMikroMoveUserManual_SM148E	PIMikroMove Operating Software (GCS-based)
GCSData_User_SM146E	GCS array data format description
PiStageEditor_SM144E	Software for managing stage databases

Contents

1 Introduction

The C-663 Mercury Step stepper motor controller is the perfect solution for cost-effective and flexible motion control applications where a precision positioner is to be controlled by a PC or PLC (programmable logic controller). The C-663 supplements the successful Mercury DC motor servo-controllers. These products are Mercury Class controllers and as such share the same command set and are internetworkable.

1.1 Overview

- RS-232 and USB host communication
- Stand-alone capability
- Network capability for multi-axis applications
- Compatible and networkable with all other Mercury class controllers

Fig. 1: C-663.11 Mercury Step controller

- Joystick port for manual control
- Non-volatile macro memory
- On-the-fly parameter changes
- TTL inputs for limit and reference switches
- Motor brake control
- Programmable I/O lines

Microstepping of 1/16 full step (for up to 6400 microsteps/rev. with PI stepper motors) provides ultra-smooth, high-resolution motion.

Multi-Axis Control, Combination of DC & Stepper Motors

The networking feature allows the user to start out with one Mercury controller and add more units later for multi-axis setups.

The C-663 Mercury Step stepper motor controller shares its GCS programming language with the well-established Mercury DC-motor controllers. Up to 16 Mercury controllers (DC and stepper) can be daisy chained and operated from one computer interface.

Flexible Automation

The C-663 offers a number of features for performing automation and handling tasks in research and industry in a very cost-effective way. Programming is facilitated by the high-level mnemonic command language with macro and compound-command functionality. Macros can be stored in the non-volatile memory for later recall.

Stand-alone capability is provided by a user-programmable autostart macro to run automation tasks at power up (no run-time computer communication required!).

For easy synchronization of motion with internal or external trigger signals, four input and four output lines are provided. A joystick can also be connected for manual control.

Command Set

C-663 Mercury Step controllers can be operated using the PI General Command Set (GCS). PI GCS allows the networking of different controller units, both for piezo-based and motorized positioning units, with minimal programming effort.

Software / Programming

In addition to the user software for setup, system optimization and operation, comprehensive LabVIEW and DLL libraries are provided which allow for easy programming and integration into your system.

The PIMikroMove user software provides graphic displays which show the system's behaviour and facilitate parameter setting.

Hardware

Easy data interchange with laptop or PC is possible via the USB interface. To facilitate industrial applications, an RS-232 interface is also standard.

The hardware of the C-663.11 is identical with that of the C-663.10. This means that it is possible to turn a C-663.11 into a C-663.10 and vice versa by installing the relevant firmware. However, this will be necessary in exceptional cases only, for example, if you have several C-663s which do not share the same command set.

1.2 Intended Use

Based on its design and realization, the C-663 Mercury Step controller is intended to drive PI stages with 2-phase stepper motors.

Observe the safety precautions given in this User Manual.

The C-663 may only be used for applications suitable according to the device specifications. Operation other than instructed in this User Manual may affect the safeguards provided.

The verification of the technical specifications by the manufacturer does not imply the validation of complete applications. In fact the operator is responsible for the process validation and the appropriate releases.

The C-663 is a laboratory apparatus as defined by DIN EN 61010. It meets the following minimum specifications for safe operation (any more stringent specifications in the technical data table are, of course, also met):

- Indoor use only
- Altitude up to 2000 m
- Temperature range 0°C to 50°C
- Max. relative humidity 80% for temperatures up to 31°C, decreasing linearly to 50% relative humidity at 40°C
- Line voltage fluctuations not greater than ±10% of the line voltage
- Transient overvoltages as typical for public power supply
  Note: The nominal level of the transient overvoltage is the standing surge voltage according to the overvoltage category II (IEC 60364-4-443).
- Degree of pollution: 2

1.3 Safety Precautions

Install and operate the C-663 Mercury Step controller only when you have read the operating instruction. Always keep this user manual next to the C-663 when using the C-663. If the user manual is lost or damaged, contact our customer service department (see p. 184). Add all information given by the manufacturer to the user manual, for example supplements or Technical Notes.

WARNING

Connect the AC power cord of the external power supply to the wall socket (100 to 240 VAC).

To disconnect the system from the supply voltage completely, remove the power plug from the wall socket.

Install the system near the AC outlet and such that the AC power plug can be reached easily.

CAUTION

Never connect a DC-motor drive to a C-663 stepper motor controller. Irreparable damage could result.

CAUTION

If operating different Mercury controllers, do not mix up the 12 V, 15 V and 24 V power supplies!

CAUTION

Never connect the RS-232-IN and USB connectors of the same controller to a PC at the same time as this can cause damage to the controller.

CAUTION

All motion of the connected motors is software controlled, and software may fail. Defective software or wrong operation of the software may result in unexpected motions which can cause equipment damage.

CAUTION

Do not enable a joystick via command when no joystick device is connected to the controller hardware. Otherwise the corresponding controller axis may start moving and could damage your application setup.

CAUTION

Wrong values of the C-663 parameters may lead to improper operation or damage of your hardware. Be careful when changing parameters.

CAUTION

If the limit switches are deactivated, the stage can run into the hard stop. This can cause damage to equipment. The limit switches are deactivated when parameter 0x32 is set to 1 ("Stage has no limit switches").

!

CAUTION

Commanding a velocity above the maximum possible for the stage will cause the motor to stall. Because stepper motors do not have position encoders, the position counter will continue to increment (it counts motor steps). The controller's motor position may not correspond with the actual motor position and this might endanger your application.

The maximum velocity depends on various influences like operating voltage, phase current setting and mechanical load. Datasheet values are for orientation only and may not work under all conditions.

Check out the maximum possible velocity for your individual application!

!

CAUTION

Because the unit is not grounded over the power supply, a grounding screw is provided at the lower left corner of the rear panel for connecting the metal case to a protective ground.

!

1.4 Unpacking

Unpack the C-663 Mercury Step controller with care. Compare the contents against the items covered by the contract and against the packing list.

The following components are included:

- C-663.11 Mercury Step controller
- Wide-range 24 V power supply (C-663.PS)
- Power-supply line-voltage cable
- RS-232 null-modem cable for PC connection (C-815.34, 3 m)
- RS-232 straight-through networking cable (C-862.CN, 28 cm)

- USB cable (type A to mini-B) for PC connection (000014651)
- Mercury product CD with all software and manuals for Mercury Class products
- MS208E User Manual for C-663 in printed form (this document)

If parts are missing or you notice signs of damage, contact PI immediately. Save all packing materials in case the product needs to be shipped again.

1.5 Accessories

The items listed below are not included but can be ordered. To order, contact your PI representative or write an e-mail to info@pi.ws.

Order Number	Description
C-815.38	Stage/motor cable, 3 m (sub-D 15 m/f)
C-862.CN2	Long straight-through networking cable for interconnecting Mercury Class controllers, 180 cm
C-819.20	Analog joystick, 2 axes
C-819.20Y	Y-cable for connecting 2 controllers to C-819.20
C-170.PB	Pushbutton box with 4 buttons and 4 LEDs
C-170.IO	Connector for I/O socket (p. 192), with cable, open end

1.6 Motion System Requirements

To start working with the C-663 Mercury Step controller, your motion system must also include the following components:

- A PC with Windows operating system (XP, Vista, 7) or Linux operating system (kernel 2.6, GTK 2.0, glibc 2.4). Note that not all software components are available for Linux PCs. See "Software Description" (p. 10) for more information.
- Communications interface to the PC:
  A free COM port on the PC or
  A free USB interface on the PC

- RS-232 null modem cable or USB cable to connect controller and host PC, or RS-232 straight-through networking cable for daisy chain connection

- A suitable stepper motor stage

- Mercury product CD with host software

1.7 Software Description

The table below lists the software tools which are on the Mercury product CD with application recommendations.

For more information see the corresponding software manuals.

Software Tool	Supported Operating System	Short Description	Recommended For
PIMikroMove	Windows	PIMikroMove permits you to start your motion system—host PC, controller and stage(s)—immediately without the need to write customized software. It offers motion-control displays and features that in many cases make it unnecessary to deal with ASCII-format commands. It also has a complete command input facility, which represents an easy way to experiment with various commands. PIMikroMove uses the GCS DLL described here to command the controller. Note that the program offers comprehensive online support.	Users who want to test the equipment before or instead of programming an application and who want to learn how to use the commands. For motor controllers, PIMikroMove offers an easy way to optimize control parameters.
GCS Library	Windows, Linux	Allows program access to the C-663 from languages like C++. The functions in the library are based on the PI General Command Set (GCS). Windows operating systems: PI_GCS2_DLL; Linux operating systems: libpi_pi_gcs2.so.x.x.x and libpi_pi_gcs2-x.x.x.a where x.x.x gives the version of the library	Recommended for users who want to use a library for their applications. The dynamic version of the library is needed by the LabVIEW driver set and by PIMikroMove.

Software Tool	Supported Operating System	Short Description	Recommended For
LabVIEW drivers	Windows, Linux	LabVIEW is a software tool (available separately from National Instruments) for data acquisition and process control. The C-663 LabVIEW software consists of a collection of virtual instrument (VI) drivers for the C-663 controller. This driver set supports the PI General Command Set (GCS). Included are Vis for GCS commands and high-level Vis for various tasks.	Users who want to use LabVIEW for programming their applications based on the GCS. See the GCS LabVIEW manual of your controller for more information.
PITerminal	Windows	PITerminal is a Windows GUI which can be used as a simple terminal with almost all PI controllers.	Users who want to send the commands of the PI General Command Set (GCS) directly.
PIStageEditor	Windows	GUI tool for adding, removing and editing stages (parameter sets) in stage parameter files (DAT files) used by the GCS library and the other host software from PI	Users who want to check or edit the content of the stage databases used by the host software

The PI host software is improved continually. It is therefore recommended that you visit the PI website (www.pi.ws) regularly to check if updated releases of the software are available for download. Updates are accompanied by information (readme files) so that you can decide if updating makes sense for your application. You need a password to check if updates are available and to download them. This password is provided on the Mercury product CD in the Releasenews PDF file in the \Manuals directory. See "Software Updates" (p. 45) for download details.

2 First Steps

For your first steps with the system, you should use PIMikroMove (see the PIMikroMove manual for more information). The following instructions describe the operation of a single C-663 controller.

During start-up, you have to select a stage type. As a result, the stage parameters will be loaded automatically from a stage database on the host PC to the controller. See "Controller Parameters" (p. 27) and "Customizing the System" (p. 65) for more information about parameter settings.

The stage selection from the database must be repeated whenever you replace the connected stage with one of another stage type. See "Parameter Databases" (p. 37) for more information.

If you wish to store your settings as default settings to the controller's non-volatile memory, see "Changing Default Parameter Values" (p. 66) for instructions.

CAUTION

!

Never connect the RS-232-IN and USB connectors of the same controller to a PC at the same time as this can cause damage to the controller.

CAUTION

!

In order to prevent damage to the stage or nonsatisfying performance, make sure that

- in the software, the stage selection corresponds to the physically connected stage type.
- the latest version of the PIStages2.dat stage database is installed. See "Installing the Software on the Host PC" (p. 23) and "Updating PIStages2.dat" (p. 45) for details.
- it is safe for the stage to move and reference the axes.

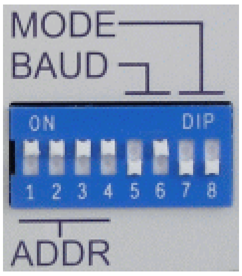
NOTE

Since the C-663 is able to store parameter values in the non-volatile memory, it may come with preset parameter values, especially when delivered with a custom stage. In PIMikroMove, the *Current stage type* column of the *Controller axes* list then already shows the name of the custom stage. In this case, do not choose any stage database entry from the list. Before you click *OK* in the *Select connected stages* window, make sure that the *Action* column of the *Controller axes* list shows *<do not change>*.

How to start operation with a single C-663:

1 Connect the single-axis stage to the "Stepper Motor only" socket of the C-663. See "Connecting Controller and Stage" (p. 22) for details.

2 Starting operation for the first time, you should use the default DIP switch settings of the C-663 which are shown in the figure below:



Controller address = 1
Baud rate = 38400 baud
Mode = Normal operation

If you want to change the default settings, see "DIP Switch Settings" (p. 21) for details.

3 Connect the C-663 to the host PC. Use either the RS-232 interface (via the "RS-232 In" socket on the controller) or the USB interface and the corresponding cable which is included in the delivery. Never connect both interfaces at the same time!

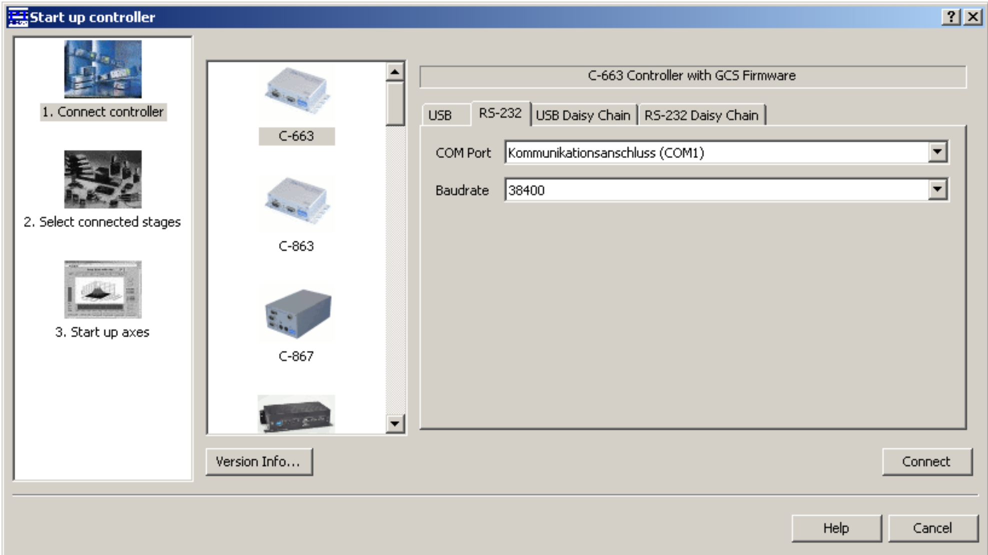
4 Connect C-663 and the included 24 V power supply (use the "15-30 VDC" socket on the C-663 rear panel).

5 Connect the power supply of the C-663 to the line power (100-240 VAC). The controller is powered on and immediately ready for operation (STA LED lights up permanently). See "Connecting Controller and Stage" (p. 22) for details.

6 Start PIMikroMove on the host PC.
See "Installing the Software on the Host PC" (p. 23) for installation details.

7 Establish a connection to the C-663 controller from PIMikroMove.
If the *Start up controller* window does not open automatically, choose *Connections* > *New…* from the menu.

The figure below shows the *Start up controller* window at the *Connect controller* step. The C-663 has address 1 and is physically connected via RS-232.



Depending on the type of interface used for the physical connection, use either the *USB, RS-232, USB Daisy Chain* or the *RS-232 Daisy Chain* tab.

RS-232:
If you use the RS-232 connection, choose the baud rate as preset by the DIP switch settings at the front panel of the controller.

USB:
When using the USB interface for the first time, two USB drivers must be installed on the host PC. These drivers are provided on the Mercury product CD in the \USB Driver directory.

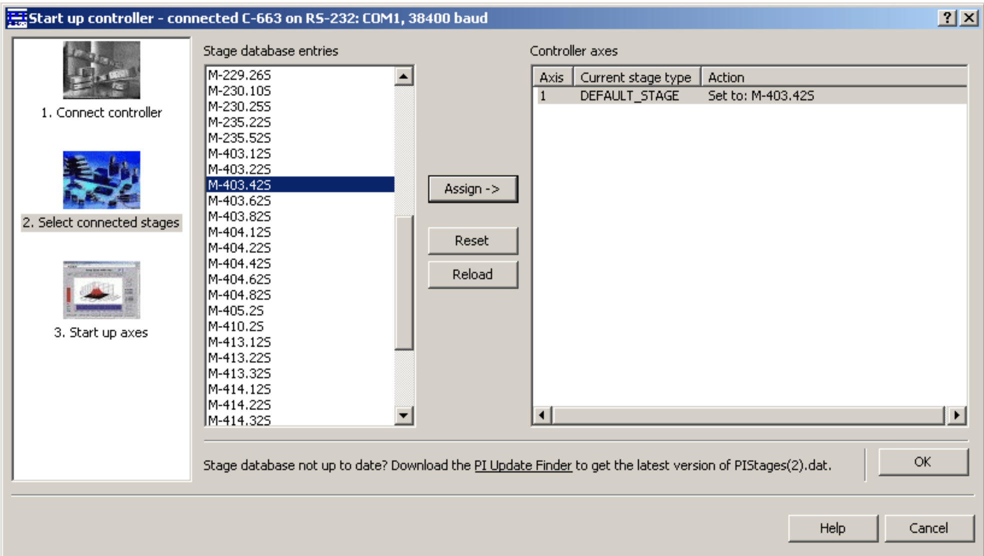
Daisy-Chain:
Note that with a daisy-chain there must be one controller with address 1. It is not required that this controller is directly connected to the host PC, i.e. this controller does not have to be the first controller of the daisy-chain. If there is no controller in a daisy-chain with address 1, an error message is displayed when you try to establish a connection.

See "Connecting Controller or Daisy-Chain Network to Host PC" (p. 25) for more information.

To establish the connection, click *Connect* in the *Start up controller* window. This will open the stage selection dialog for the next step. If it does not open automatically, click *Select connected stages* in the window pane on the left.

8 Choose the stage type to be connected.

The figure below shows the *Start up controller* window with the *Select connected stages* step.



If the stage selection dialog does not open automatically, choose *Select connected stages…* from the *C-663* menu in the PIMikroMove main window (e.g. *C-663 (COM1) > Select connected stages…)*.

Check if the *Current stage type* in the *Controller axes* pane matches your stage. There are two possibilities:

a) The *Current stage type* does not match your stage:
In this case, choose the appropriate stage type from the *Stage database entries* list and click the *Assign* button.
The selected stage type is now displayed in the *Controller axes* pane.

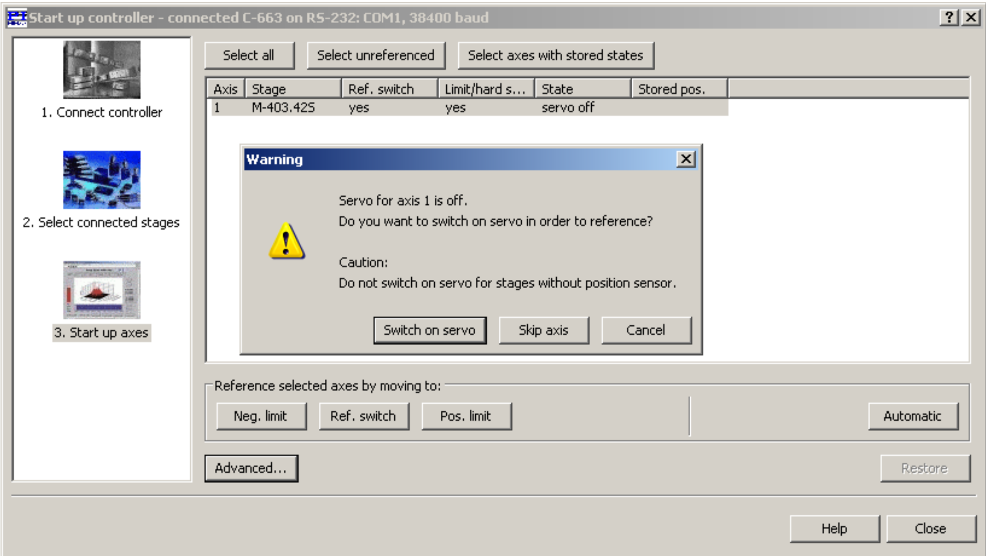
b) The *Current stage type* matches your stage:
Make sure that *<do not change>* is displayed in the *Action* column of the *Controller axes* pane. Do not change the settings.

9 To accept the stage selection and to close the *Select connected stages* dialog, click the *OK* button. (The choice can later be changed with *Select connected stages…* from the *C-663* menu in the PIMikroMove main window.)
The *Start up axes* dialog opens where you can reference your stage.

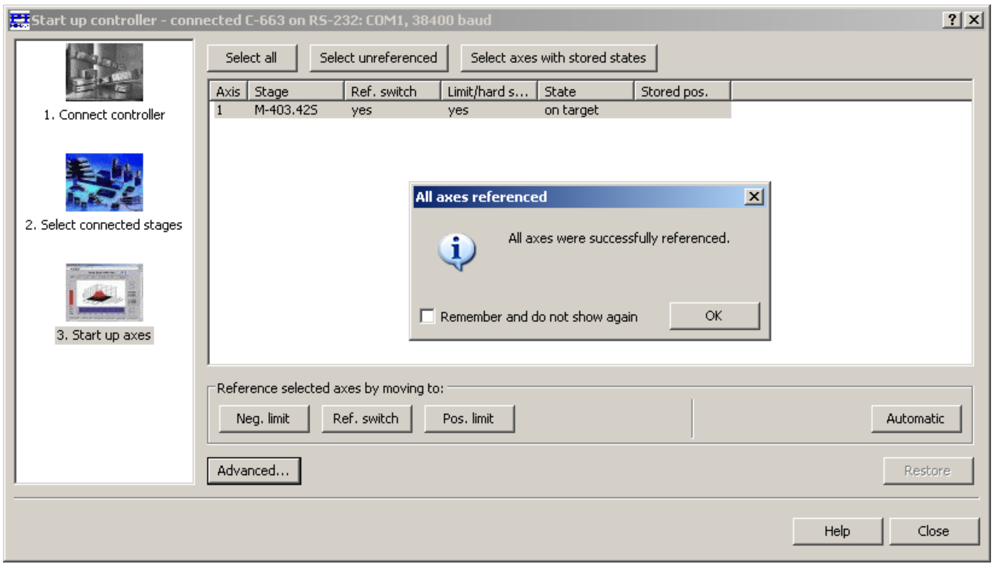
10 Start a reference move for the axis.

The controller cannot know the absolute position of an axis upon startup. Reference and/or limit switches in the stage can be used to obtain absolute position information.

The figures below show the *Start up controller* window with the *Start up axes* step.



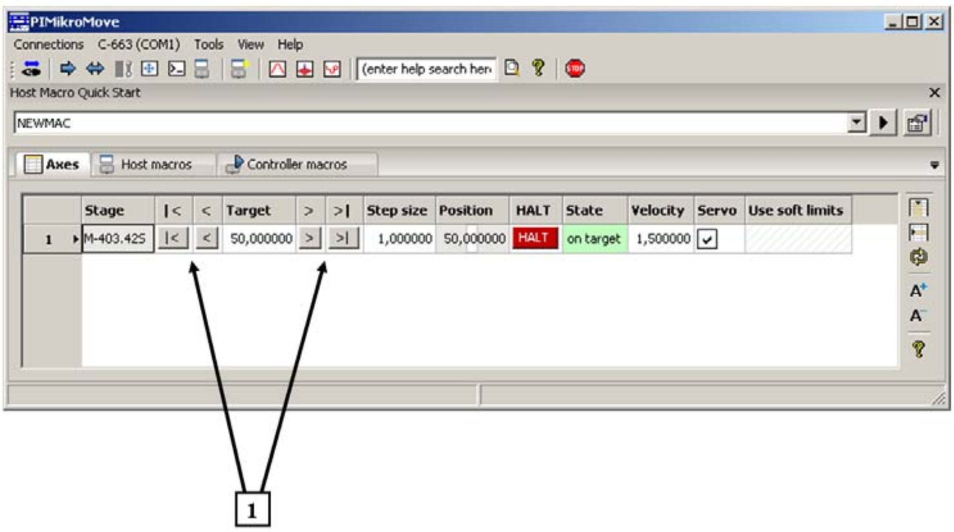
To start the reference move, choose a referencing method by clicking the *Neg. limit*, *Ref. switch* or *Pos. limit* button. If you have just powered on your controller, a dialog window will open informing you that servo is switched off. In this case, click the *Switch on servo* button. Since the C-663 has no servo loop, this will switch on the motor of your stage.



When referencing has been completed successfully, click *OK* > *Close*. The PIMikroMove main window will open. See "Referencing" (p. 37) for more information.

11 Start some test moves of the axis:
For example, perform a step of a predefined size by clicking on the associated arrow buttons for the axis.

The figure below shows the PIMikroMove main window with the *Axes* tab where you can start axis motion.



[1] Arrow buttons causing motion

See "Working with Controller Macros" (p. 59) for how to store macros in the C-663's non-volatile memory for later recall and

"Joystick Control" (p. 56) for manual control by a joystick connected to the C-663.

3 Details of Operation

3.1 Installing the C-663

!

CAUTION

Place the system in a location with adequate ventilation to prevent internal heat build-up. Allow at least 10 cm (4 inches) clearance from the top and the rear of the unit and 5 cm (2 inches) from each side.

The C-663 can be used as desktop device or mounted on a base in any orientation. If you want to mount the C-663 on a base, see "Mounting Hole Pattern" (p. 188).

Because grounding is not assured over the power connection, the C-663 chassis must be connected to a protective ground via the labeled screw on the rear panel.

3.2 Front and Rear Panel Elements

3.2.1 Front Panel Elements

Figure 1: C-663 front panel

Name	Function
RS-232 In	Serial connection to host PC or to previous controller in a daisy-chain network. See "RS-232 In and RS-232 Out Sockets" (p. 190) for pinout.
RS-232 Out	Serial connection to next controller in a daisy-chain network. See "RS-232 In and RS-232 Out Sockets" (p. 190) for pinout.
STA LED (green)	Power on and ready indicator. When power is applied to the controller, the LED will glow for normal operation. Stays off when the C-663 is in firmware update mode.
ERR LED (red)	Error indicator; when LED lights up, error code is non-zero and can be queried and cleared using the ERR? command (p. 102). Stays off when the C-663 is in firmware update mode.
	Universal Serial Bus (USB Mini-B (m) socket) for connection to host PC. See "USB Socket" (p. 190) and "USB Interface" (p. 25) for more information.
Mode, Baud, Addr	8-bit DIP switch, sets controller address, RS-232 baud rate and operating mode (normal operation / firmware update) of the unit. See "DIP Switch Settings" (p. 21) for details.

3.2.2 DIP Switch Settings

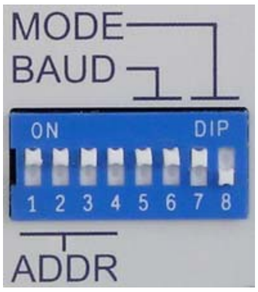


Figure 2:
Slider up = ON,
slider down = OFF

Name	Function
Addr (switches 1 to 4)	Controller address (1 to 16)
Baud (switches 5 and 6)	Baud rate (9600, 19200, 38400 or 115200)
Mode (switch 8)	Operating mode (normal operation or firmware update)

Note: Switch 7 has no function.

Factory settings are shown in bold in the tables below.

Address	SW1	SW2	SW3	SW4
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

Mode	SW8
Firmware update	ON
Normal	OFF

Baud Rate*	SW5	SW6
9600	ON	ON
19200	ON	OFF
38400	OFF	ON
115200	OFF	OFF

*Other settings are fixed at 8 data, 1 stop, no parity; internal buffers are used so there is no handshake required.

3.2.3 Rear Panel Elements

Figure 3: Rear Panel of C-663

CAUTION

Never connect a DC-motor drive to a C-663 stepper motor controller. Irreparable damage could result.

Name	Function
15-30 VDC	Barrel connector for power supply. See "15-30 VDC Socket" (p. 194) for pinout.
I/O	Mini DIN 9-pin connector, provides digital I/O and analog input lines. See "I/O Socket" (p. 191) for pinout.
Joystick	Mini DIN 6-pin connector for analog joystick (input). See "Joystick Socket" (p. 193) for pinout.
Stepper Motor only	Sub-D 15(f) socket for motor/stage connection (I/O): Stepper Motor only! See "Motor Connector" (p. 189) for pinout.
	Screw (and washer) for protective ground connection

3.3 Connecting Controller and Stage

CAUTION

Never connect a DC-motor drive to a C-663 stepper motor controller. Irreparable damage could result.

Connect the single-axis stage to the "Stepper Motor only" socket of the C-663. The C-663 is adapted to the connected stage type using controller parameter settings. See "First Steps" (p. 12) and "Controller Parameters" (p. 27) for more information.

3.4 Supply Power Connection

The C-663 comes with a 24 V wide-range-input power supply that can be used with line voltages from 100 VAC to 240 VAC at 50 or 60 Hz.

To power on the C-663, proceed as follows:

1. Because grounding is not assured over the power connection, connect the C-663 chassis to a protective ground via the labeled screw on the rear panel (see figure below).

2. Connect the included wide-range power supply to the "15-30 VDC" connector of the C-663.

3. Connect the AC power cord of the power supply to the wall socket. When the green STA LED glows, the C-663 is ready for normal operation. If the STA LED does not glow, check the DIP switch settings on the C-663 front panel: All LEDs stay off when the C-663 is in firmware update mode (DIP switch 8 in ON position).

3.5 Installing the Software on the Host PC

Windows operating systems:

1. Insert the Mercury product CD in your host PC.

2. If the Setup Wizard does not open automatically, start it from the root directory of the CD by opening the *setup.exe* file.

3. Follow the on-screen instructions and select the “typical” installation. Typical components are LabView drivers, GCS DLL, PIMikroMove.

Linux operating systems:

1. Insert the Mercury product CD in the host PC.

2. Open a terminal and go to the /linux directory on the Mercury product CD.

3. Log in as superuser (root).

4 Start the install script with ./INSTALL
Keep in mind the case sensitivity of Linux when typing the command.

5 Follow the on-screen instructions. You can choose the individual components to install.

If the installation fails, make sure you have installed the kernel header files for your kernel.

The PIStages2.dat stage database file needed by the host software is installed in the ...\PI\GCSTranslator directory. In that directory, also the *Prefix*UserStages2.dat database will be located which is created automatically the first time you connect stages in the host software (i.e. the first time the VST? or CST functions of the GCS library are used).
The location of the PI directory is that specified upon installation, usually in C:\Documents and Settings\All Users\Application Data (Windows XP) or C:\ProgramData (Windows Vista and Windows 7). If this directory does not exist, the EXE file that needs the stage databases will look in its own directory. Note that in PIMikroMove, you can use the *Version Info* entry in the controller menu or the *Search for controller software* entry in the *Connections* menu to identify the GCSTranslator path.

It is strongly recommended to always use the latest version of the PIStages2.dat stage database. The content of this database is maintained continually, e.g. new stage types are added, and parameters are optimized. Therefore the version on the Mercury product CD may be out of date. The latest PIStages2.dat file is available for download on the PI website (www.pi.ws), see "Updating PIStages2.dat" (p. 45) for details.

For an overview of the host software provided see "Software Description" (p. 10).

The PI host software is improved continually. It is therefore recommended that you visit the PI website (www.pi.ws) regularly to check if updated releases of the software are available for download. Updates are accompanied by information (readme files) so that you can decide if updating makes sense for your application. You need a password to check if updates are available and to download them. This password is provided on the Mercury product CD in the Releasenews PDF file in the \Manuals directory. See "Software Updates" (p. 45) for download details.

3.6 Connecting Controller or Daisy-Chain Network to Host PC

!

CAUTION

Never connect the RS-232-IN and USB connectors of the same controller to a PC at the same time as this can cause damage to the controller.

Use either the RS-232 or the USB interface to connect the C-663 or a C-663 daisy chain network to the host PC.

Up to 16 C-663 controllers can be controlled from a single host computer interface. The RS-232 output stages of some PCs may not be capable of driving more than 6 units; if this is a problem use USB to interface with the PC. Interconnect any additional controllers being networked to the network with straight-through RS-232 cables chaining off the RS-232 OUT connector of the controller connected to the PC (one straight-through RS-232 cable (C-862.CN) comes with each C-663 controller).

3.6.1 USB Interface

The first time you connect via the USB interface, ensure that you are logged on to the PC as a user having administrator rights. After the C-663 is powered on, a message will appear saying that new hardware has been detected. Follow the on-screen instructions and insert the Mercury product CD again. The required hardware drivers are found in the \USB Driver directory.

The USB drivers will make the USB interface appear to all software on the host PC as a new COM port. That port will be present only when the controller is connected via USB and powered on. Depending on the way the connection between C-663 and host PC is established in the host software, it may be possible to select the baud rate of that PC COM port. Make sure that the selection corresponds to the baud rate settings of the C-663 (made via the DIP switches on the C-663 front panel).

3.6.2 Baud Rate Settings

The baud rate can be set to one of 9600, 19200, 38400 and 115200 using the Baud DIP switches on the C-663 front panel, see "DIP Switch Settings" (p. 21) for details. All controllers in a daisy chain network must be set to the same baud rate. Other communication settings are fixed at 8 data, 1 stop, no parity; internal buffers are used so there is no handshake required.

3.6.3 Address Settings

The controller address of the C-663 can be set with the Addr DIP switches on the front panel, see "DIP Switch Settings" (p. 21) for details. Possible controller addresses are in the range of 1 to 16, address 1 is default. The host PC always has the address 0. See "Target and Sender Address" (p. 75) for more information about the format of the command line.

In a daisy chain network, each C-663 must have a unique controller address, and one controller of the daisy-chain must have address 1.

The communication on the interface is between the host computer and a specifically addressed controller in the chain. With the broadcast address 255, all controllers can be addressed at the same time, but no reports are displayed on the host PC.

NOTES

Except when making the required hardware settings, you have to deal with addresses only if the connection between C-663 and host PC is done via a terminal program without any intervening software layers (e.g. DLLs). With PITerminal, this is the case if the connection is established by clicking the *Connect…* button.

3.7 Controller Parameters

3.7.1 Parameter Descriptions and Handling

The hardware basics of the connected stage and the required control settings are mirrored in controller parameters. The parameter values have to be adjusted properly before initial operation of a stage.

With HPA? (p. 113) you can obtain a list of all available parameters with information about each (e.g. short descriptions). The volatile and non-volatile memory parameter values can be read with the SPA? (p. 145) or SEP? (p. 140) commands, respectively.

Using the "general" modification commands SPA, RPA, SEP and WPA, parameters can be changed in volatile memory (SPA (p. 142), RPA (p. 136)) or in non-volatile memory (SEP (p. 139), WPA (p. 160)). It is recommended that any modifications be first made with SPA, and when the controller runs well, saved using WPA. In addition to the "general" modification commands, there are commands which change certain specific parameters in volatile memory (see table below). For information on how to change parameter values using PI software, see "Changing Default Parameter Values" (p. 66).

When a firmware update is available which introduces new parameters (see the documentation that comes with the update), then those new parameters must be set to initial values by a special command. See "Firmware Updates" (p. 47) for more information.

!

CAUTION

Wrong values of the C-663 parameters may lead to improper operation or damage of your hardware. Be careful when changing parameters.

The interrelation of the hardware-dependent parameters 0x15, 0x16, 0x17, 0x2F and 0x30 is described in "Travel Range Adjustment" (p. 68).

For further details regarding parameters see "Trajectory Generation" (p. 53).

Values stored in non-volatile memory are default settings, so that the system can be used in the desired way immediately.

Parameter ID (hexa-decimal)	Data Type	Password for Writing to Non-Volatile Memory	Parameter Description	Possible Values/Notes
0xA	FLOAT	100	Maximum velocity (user unit/s)	Gives the maximum value for parameter 0x49.
0xB	FLOAT	100	Current acceleration (user unit/s^2), also changed by ACC command (p. 85)	Gives the current acceleration, limited by parameter 0x4A
0xC	FLOAT	100	Current deceleration (user unit/s^2), also changed by DEC command (p. 95)	Gives the current deceleration, limited by parameter 0x4B
0xE	INT	100	Numerator of the counts-per-physical-unit factor	1 to 1,000,000 for each parameter. The counts-per-physical-unit factor determines the "user" unit for motion commands. When you change this factor, all other parameters whose unit is based on the "user" unit are adapted automatically, e.g. velocity and parameters regarding the travel range.
0xF	INT	100	Denominator of the counts-per-physical-unit factor	
0x13	INT	100	Rotary stage?	0 = no 1 = yes This parameter has no effect on the firmware but is relevant to PIMikroMove where it is used to indicate which kind of motion is allowed.
0x14	INT	100	Stage has a built-in reference switch	1 = the stage has a built-in reference switch (line on the Sub-D 15 (f) motor connector), else 0
0x15	FLOAT	100	MAX_TRAVEL_RANGE_POS The maximum travel in positive direction (user unit)	"Soft limit", based on the home (zero) position. If the soft limit is smaller than the position value for the positive limit switch (which is given by the sum of the parameters 0x16 and 0x2F), the positive limit switch cannot be used for referencing. Can be negative.

Parameter ID (hexa-decimal)	Data Type	Password for Writing to Non-Volatile Memory	Parameter Description	Possible Values/Notes
0x16	FLOAT	100	VALUE_AT_REF_POS The position value at the reference position (user unit)	The position value which is to be set when the mechanical system performs a reference move to the reference switch. Is furthermore used to calculate the position values to be set after reference moves to the limit switches, even if no reference switch is present in the mechanical system.
0x17	FLOAT	100	DISTANCE_REF_TO_N_LIM The distance between reference switch and negative limit switch (user unit)	Represents the physical distance between the reference switch and the negative limit switch integrated in the mechanical system. When the mechanical system performs a reference move to the negative limit switch, the position is set to the difference of VALUE_AT_REF_POS and DISTANCE_REF_TO_N_LIM.
0x18	INT	100	Axis limit mode	0 = positive limit switch active high (pos-HI), negative limit switch active high (neg-HI) 1 = positive limit switch active low (pos-LO), neg-HI 2 = pos-HI, neg-LO 3 = pos-LO, neg-LO
0x1A	INT	100	Has brake	0 = no 1 = yes
0x2F	FLOAT	100	DISTANCE_REF_TO_P_LIM The distance between reference switch and positive limit switch (user unit)	Represents the physical distance between the reference switch and the positive limit switch integrated in the mechanical system. When the mechanical system performs a reference move to the positive limit switch, the position is set to the sum of VALUE_AT_REF_POS and DISTANCE_REF_TO_P_LIM.

Parameter ID (hexa-decimal)	Data Type	Password for Writing to Non-Volatile Memory	Parameter Description	Possible Values/Notes
0x30	FLOAT	100	MAX_TRAVEL_RANGE_NEG The maximum travel in negative direction (user unit)	"Soft limit", based on the home (zero) position. If the soft limit is larger than the position value for the negative limit switch (which is given by the difference of the parameters 0x16 and 0x17), the negative limit switch cannot be used for referencing. Can be negative.
0x31	INT	100	Invert the reference	This parameter can be used to invert either the reference sensor signal or a digital input which is used instead of the reference sensor (see parameter 0x5c). 0 = no 1 = yes
0x32	INT	100	Stage has limit switches	This parameter enables / disables the stopping of the motion at the built-in limit switches. 0 = yes (stage has built-in limit switches; lines on the Sub-D 15 (f) motor connector) 1 = no
0x3C	CHAR	100	Stage name	Default is "DEFAULT_STAGE"
0x40	INT	100	Holding current (mA)	To hold the position when the stage is on target, the operating current is reduced to the holding current. Depending on the application, the holding current should be set to 20 % - 30 % of the specified maximum operating current to reduce motor warming.
0x41	INT	100	Operating current (mA)	This current is applied while the motor of the stage is moving.
0x42	INT	100	Holding current delay (ms)	Delay in milliseconds between operating and holding current (see parameters 0x40 and 0x41).

Parameter ID (hexa-decimal)	Data Type	Password for Writing to Non-Volatile Memory	Parameter Description	Possible Values/Notes
0x47	INT	100	Default direction for reference	0 = detect automatically, 1 = start in negative direction, 2 = start in positive direction
0x49	FLOAT	100	Current velocity (user unit/s) also changed by VEL command (p. 158)	Gives the current velocity, limited by parameter 0xA
0x4A	FLOAT	100	Maximum acceleration (user unit/s^2)	Gives the maximum value for parameter 0xB
0x4B	FLOAT	100	Maximum deceleration (user unit/s^2)	Gives the maximum value for parameter 0xC
0x50	FLOAT	100	Velocity for reference move (user unit/s)	Gives the maximum velocity to be used for reference moves with FRF, FPL, FNL; if set to 0, reference moves are not possible
0x5C	INT	100	DIO as REF	You can use a digital input instead of the reference sensor as source of the reference signal for the FRF or FED command: 0 = Reference sensor 1 = Digital input 1 2 = Digital input 2 3 = Digital input 3 4 = Digital input 4 Ensure that the digital input signal you use for referencing only switches once across the entire travel range from low to high. If your signal switches from high to low, you can invert it using the parameter 0x31. The selected digital input line is used for referencing irrespective of the setting of parameter 0x14 (“Stage has a built-in reference switch”).

Parameter ID (hexa-decimal)	Data Type	Password for Writing to Non-Volatile Memory	Parameter Description	Possible Values/Notes
0x5D	INT	100	DIO as NLIM	You can use digital input lines as sources of the negative limit signal (and hence also for FNL or FED). This parameter is bit-mapped. The values are as follows: 0 = Negative limit switch (default) 1 = Digital input 1 (bit 0) 2 = Digital input 2 (bit 1) 4 = Digital input 3 (bit 2) 8 = Digital input 4 (bit 3) While the built-in limit switches of the stage are processed only if enabled with parameter 0x32 (“stage has limit switches”), the setting of parameter 0x32 has no influence on the usage of activated digital input lines. To activate several inputs, sum up the values accordingly. For example, to activate the digital inputs 1, 3 and 4, set this parameter to 13. When referencing, only one signal source can be active. Ensure that the digital input signal you use for referencing only switches once across the entire travel range from low to high. If your signal switches from high to low, you can invert it using the parameter 0x5F (Invert DIO-NLIM).

Parameter ID (hexa-decimal)	Data Type	Password for Writing to Non-Volatile Memory	Parameter Description	Possible Values/Notes
0x5E	INT	100	DIO as PLIM	You can use digital input lines as sources of the positive limit signal (and hence also for FPL or FED). This parameter is bit-mapped. The values are as follows: 0 = Positive limit switch (default) 1 = Digital input 1 (bit 0) 2 = Digital input 2 (bit 1) 4 = Digital input 3 (bit 2) 8 = Digital input 4 (bit 3) While the built-in limit switches of the stage are processed only if enabled with parameter 0x32 (“stage has limit switches”), the setting of parameter 0x32 has no influence on the usage of activated digital input lines. To activate several inputs, sum up the values accordingly. For example, to activate the digital inputs 1, 3 and 4, set this parameter to 13. When referencing, only one signal source can be active. Ensure that the digital input signal you use for referencing only switches once across the entire travel range from low to high. If your signal switches from high to low, you can invert it using the parameter 0x60 (Invert DIO-PLIM).

Parameter ID (hexa-decimal)	Data Type	Password for Writing to Non-Volatile Memory	Parameter Description	Possible Values/Notes
0x5F	INT	100	Invert DIO-NLIM	The polarity of the digital input serving as negative limit signal can be inverted (active high or active low). This parameter is bit-mapped. The values are as follows: 0 = No digital input is inverted (default) 1 = Digital input 1 inverted (bit 0) 2 = Digital input 2 inverted (bit 1) 4 = Digital input 3 inverted (bit 2) 8 = Digital input 4 inverted (bit 3) To invert several inputs, sum up the values accordingly. For example, to invert the digital inputs 1, 3 and 4, set this parameter to 13.
0x60	INT	100	Invert DIO-PLIM	The polarity of the digital input serving as positive limit can be inverted (active high or active low). This parameter is bit-mapped. The values are as follows: 0 = No digital input is inverted (default) 1 = Digital input 1 inverted (bit 0) 2 = Digital input 2 inverted (bit 1) 4 = Digital input 3 inverted (bit 2) 8 = Digital input 4 inverted (bit 3) To invert several inputs, sum up the values accordingly. For example, to invert the digital inputs 1, 3 and 4, set this parameter to 13.

Parameter ID (hexa-decimal)	Data Type	Password for Writing to Non-Volatile Memory	Parameter Description	Possible Values/Notes
0x61	INT	100	Invert Joystick	Use this parameter to invert the direction of motion for joystick-controlled axes. 0 = not inverted (default) When a joystick axis is moved, for example, to the right, the connected stage axis moves in the positive direction of motion. 1 = inverted When a joystick axis is moved, for example, to the right, the connected stage axis moves in the negative direction of motion.
0x63	FLOAT	100	Distance between limit and hard stop	Defines the maximum braking distance when referencing. The velocity during referencing is calculated based on this value and the set deceleration. Usually, this value does not need to be changed.
0x72	INT	100	Ignore macro error	Determines whether an error during controller macro execution causes the macro to stop 0: stop on error 1: ignore error

Parameter ID (hexa-decimal)	Data Type	Password for Writing to Non-Volatile Memory	Parameter Description	Possible Values/Notes
0x07000601	CHAR	100	Axis unit	The axis unit is “MM”, for example, if the counts per unit ratio defined by the parameters 14 and 15 converts the counts to millimeters. It is used by the host software for display purposes. Example: 1 count = 100 nm Counts per unit: 10000:1 Axis unit: mm 1 count = 0.254 mm Counts per unit: 100:1 Axis unit: inch Rotary stages use “deg” as unit. This parameter has no effect on the firmware (like parameter 0x13 "Rotary stage?").
0x07000000	FLOAT	100	Travel range minimum (user unit)	Not used.
0x07000001	FLOAT	100	Travel range maximum (user unit)	Not used.

3.7.2 Parameter Databases

NOTE

The GCS-based host software from PI uses multiple databases for stage parameters:

- PIStages2.dat contains parameter sets for all standard stages from PI and is automatically installed on the host PC with the setup. It cannot be edited; should changes in the file become necessary, you must obtain a new version from PI and install it on your host PC.

- *Prefix*UserStages2.dat allows you to create and save your own stages. This database is created the first time you connect stages in the host software (i.e. the first time the VST? or CST functions of the GCS library are used). *Prefix* depends on the GCS library used, e.g. if your controller uses the PI GCS 2 library, *Prefix* will be *PI*. There can be one file of this type for each different GCS library.

When you are working with the host software from PI, you can select the suitable stage parameter set from one of the databases. The host software will then send the appropriate parameter values to the controller's volatile memory.

In case you want to operate a stage with other parameters than stated in the PIStages2.dat or if you have a customized stage, add a new stage parameter set to PIUserStages2.dat (the host software for the C-663 uses the PI GCS 2 library). For details see "Adding Stages to User DAT Files" (p. 65).

For further information, refer to the PIMikroMove manual, the PIStageEditor manual or the PI GCS library manual.

3.8 Referencing

Because the signals (motor steps) used for position determination provide only relative motion information, the controller cannot know the absolute position of an axis upon startup. This is why a referencing procedure is required before absolute target positions can be commanded and reached.

3.8.1 Reference Mode

The current reference mode setting of the controller (ask with RON? (p. 136)) determines how referencing can be performed. By default, a reference move must be performed (p. 38), but it is also possible to set absolute positions manually (p. 39). To switch between the two reference modes, use the RON command (p. 135).

3.8.2 Perform a Reference Move

When the reference mode is set to "1" (factory default), referencing is done by performing a reference move with FRF (p. 108), FPL (p. 107) or FNL (p. 105).

Neither relative nor absolute targets can be commanded as long as referencing has not been performed successfully.

FRF requires that there is a reference signal (ask with TRS? (p. 155) for a built-in reference switch), and FPL and FNL require that there are limit signals (ask with LIM? (p. 123) for built-in limit switches). The built-in limit switches of the stage can only be used for reference moves if the travel range is not reduced by soft limits, see "Travel Range Adjustment" (p. 68) for more information.

For best repeatability, always reference in the same way.

The FRF command always approaches the reference switch from the same side, no matter where the axis is when it is sent.

With the C-663, the motor of the stage must be switched on with SVO (p. 149) before reference moves can be started with FRF, FNL or FPL.

NOTES

You can use the digital input lines as source of the reference signal, the negative limit signal or the positive limit signal. See parameters 0x5C (p. 31), 0x5D (p. 32) and 0x5E (p. 33) for details.

The digital input line selected as source of the reference signal is used for referencing irrespective of the setting of parameter 0x14 (“Stage has a built-in reference switch”).

While the built-in limit switches of the stage are processed only if enabled with parameter 0x32 (“stage has limit switches”), the setting of parameter 0x32 has no influence on the usage of activated digital input lines.

3.8.3 Set Absolute Position

When the reference mode is set to "0", referencing is done by entering an absolute position value using the POS command (p. 133).

NOTES

Only relative targets but no absolute targets can be commanded as long as referencing has not been performed successfully.

If the controller is given an incorrect position with POS, the axis can run into a limit switch and will not be able to move away from the switch due to the travel range limits given by the MAX_TRAVEL_RANGE_POS parameter (ID 0x15; ask with TMX? (p. 152)) and the MAX_TRAVEL_RANGE_NEG parameter (ID 0x30; ask with TMN? (p. 152)).

3.9 Using Trigger Input and Output

3.9.1 How to Use Digital I/O Lines—Overview

It is possible to trigger external devices, to program start/stop actions in macros and to perform reference moves using the digital I/O lines of the C-663. See "I/O Socket" (p. 191) for the lines and pinout. The number of digital I/O lines available on the C-663 can be queried using the TIO? command (p. 151).

You can set the states of the Output 1 to Output 4 lines (TTL, active high) using the DIO command (p. 96), e.g. to trigger other devices. The lines can be set individually or all at once according to a bit pattern. Furthermore, you can program the Output 1 to Output 4 lines using the CTO command (p. 90) (trigger configuration) and the TRO command (p. 153) (trigger enabling/disabling). See "Configuring Trigger Output" for examples.

The states of the Input 1 to Input 4 lines (TTL, active high) can be queried with the DIO? command (p. 97). These lines can be used to stop macros and to trigger certain actions in macros via the MEX command (p. 128) or the WAC command (p. 159), respectively. See "Working with Controller Macros" (p. 59) for an example. You can also use the digital input lines as source of the reference signal, the negative limit signal or the positive switch signal. See parameters 0x5C (p. 31), 0x5D (p. 32) and 0x5E (p. 33) for details.

3.9.2 Configuring Trigger Output

You can program the digital output lines of the C-663 to trigger other devices using the CTO command (p. 90) (in combination with the TRO (p. 153) command).

The format of the CTO command is as follows (i.e. one setting can be made per command):

CTO <TrigOutID> <CTOPam> <Value>

The following trigger modes are supported by the C-663:

- 0 = "Position Distance"; a trigger pulse is written whenever the axis has covered a given distance. Optionally, values for StartThreshold and StopThreshold can be defined to enable the trigger output for a limited position range and a certain direction of motion only (negative or positive). When StartThreshold and StopThreshold are set to the same value, they will not be used. See "Example—"Position Distance" Trigger Mode"

- 2 = "OnTarget"; the on-target status of the selected axis is written to the selected trigger output line (this status can also be read with the ONT? command). See "Example—"On Target" Trigger Mode" (p. 42)

- 6 = "InMotion"; the selected trigger line is active as long as the selected axis is in motion. See "Example—"In Motion" Trigger Mode" (p. 42)

- 7 = "Position+Offset"; the first trigger pulse is written when the selected axis has reached the position given by the <TriggerPosition> parameter of the CTO command. The next trigger pulses each are written when the axis position equals the sum of the last valid trigger position and the increment value given by the <TriggerStep> CTO parameter. Trigger output ends when the axis position exceeds the value given by the <StopThreshold> CTO parameter. The sign of the <TriggerStep> value determines for which direction of motion trigger pulses are to be output. See "Example—"Position+Offset" Trigger Mode" (p. 43)

To select the mode, set <CTOPam> to 3 and <Value> to the code of the mode; default selection is "Position Distance" (0).

Furthermore, it is possible to select the signal polarity for the digital output line (active high / active low). See "Example—Polarity Setting" (p. 44).

The following examples can be reproduced using the command entry facilities of PIMikroMove or PI Terminal.

Example—"Position Distance" Trigger Mode

The "Position Distance" trigger mode is designed for scanning applications. A trigger pulse is written whenever the axis has covered the distance given by the <TriggerStep> parameter of the CTO command. The pulse width is one controller cycle (50 µs).

The default unit of <TriggerStep> is mm (depends on the settings of parameters 0xE and 0xF, see "Parameter Descriptions and Handling" (p. 27) for more information).

Send a sequence of the following commands for the digital output line (<TrigOutID>) which is to be used for trigger output (the order of the commands is irrelevant):

CTO <TrigOutID> 2 *Axis*
CTO <TrigOutID> 3 *Triggermode*
CTO <TrigOutID> 1 *Stepsize*

Example 1: A pulse on the digital output line 1 is to be generated whenever 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

Optionally, start and stop values can be set with the <StartThreshold> and <StopThreshold> parameters of the CTO command. They enable the trigger output for a limited position range and a certain direction of motion only (positive or negative). When <StartThreshold> and <StopThreshold> are set to the same value, they will not be used. Note: Should the motion direction be reversed before the axis position has reached the stop threshold, trigger pulses will continue to be generated.

Send a sequence of the following commands for the digital output line (<TrigOutID>) which is to be used for trigger output (the order of the commands is irrelevant):

CTO <TrigOutID> 2 *Axis*
CTO <TrigOutID> 3 *Triggermode*
CTO <TrigOutID> 1 *Stepsize*
CTO <TrigOutID> 8 *Startposition*
CTO <TrigOutID> 9 *Stopposition*

Example 2: A pulse on the digital output line 1 is to be generated whenever axis 1 of the stage has covered a distance of 0.1 µm, as long as axis 1 moves in positive direction in the range of 0.2 µm to 0.55 µm (start threshold < stop threshold). Send:

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

Example 3: A pulse on the digital output line 1 is to be generated whenever axis 1 of the stage has covered a distance of 0.1 µm, as long as axis 1 moves in negative direction in the range of 0.55 µm to 0.2 µm (start threshold > stop threshold). Send:

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

Example—"On Target" Trigger Mode

With the "On Target" trigger mode, the on-target status of the selected axis is written to the selected trigger line. It is the same on-target status flag which can also be read by the ONT? command (p. 133), #4 (p. 82) or the SRG? command (p. 146). The on-target status becomes true when the trajectory has finished.

Send a sequence of the following commands for the digital output line (<TrigOutID>) which is to be used for trigger output (the order of the commands is irrelevant):

CTO <TrigOutID> 2 *Axis*
CTO <TrigOutID> 3 *Triggermode*

Example: The On-Target status flag of axis 1 is to be written to the digital output line 1. Send:

CTO 1 2 1
CTO 1 3 2

Example—"In Motion" Trigger Mode

With the "In Motion" trigger mode, the selected trigger line is active as long as the selected axis is in motion.

You can use #5 (p. 83), #4 (p. 82) or the SRG? command (p. 146) to check if an axis is in motion (if so, bit 14 of status register 1 of the axis is set).

Send a sequence of the following commands for the digital output line (<TrigOutID>) which is to be used for trigger output (the order of the commands is irrelevant):

CTO <TrigOutID> 2 *Axis*
CTO <TrigOutID> 3 *Triggermode*

Example: The digital output line 1 is to be active as long as axis 1 of the stage is in motion. Send:

CTO 1 2 1
CTO 1 3 6

Example—"Position + Offset" Trigger Mode

With the "Position+Offset" trigger mode, the first trigger pulse is written when the axis has reached the position given by the <TriggerPosition> parameter of the CTO command. The next trigger pulses each are written when the axis position equals the sum of the last valid trigger position and the increment value given by the <TriggerStep> CTO parameter. Trigger output ends when the axis position exceeds the value given by the <StopThreshold> CTO parameter. The sign of the <TriggerStep> value determines for which direction of motion trigger pulses are to be output. The pulse width is one controller cycle (50 µs).

The default unit of <TriggerPosition>, <TriggerStep> and <StopThreshold> is mm (depends on the settings of parameters 0xE and 0xF, see "Parameter Descriptions and Handling" (p. 27) for more information).

Trigger processing is done by the DSP of the C-663.

Send a sequence of the following commands for the digital output line (<TrigOutID>) which is to be used for trigger output (the order of the commands is irrelevant):

CTO <TrigOutID> 2 *Axis*
CTO <TrigOutID> 3 *Triggermode*
CTO <TrigOutID> 1 *Stepsize*
CTO <TrigOutID> 10 *Triggerposition*
CTO <TrigOutID> 9 *StopThreshold*

Example 1: On the digital output line 1, the first trigger pulse is to be output if the absolute position of axis 1 is 1.5 mm. Afterwards, a pulse is to be generated on this line whenever axis 1 has covered a distance of 0.1 µm in positive direction. The last trigger pulse is to be output when the absolute position of the axis 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

Example 2: On the digital output line 2, the first trigger pulse is to be output if the absolute position of axis B is 0.4 mm. Afterwards, a pulse is to be generated on this line whenever axis B has covered a distance of 1 µm in negative direction. The last trigger pulse is to be output when the absolute position of the axis 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

NOTE

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 controller cycle of the C-663.

Example—Polarity Setting

It is possible to select the signal polarity (active high = 1, default / active low = 0) for the digital output line which is to be used for trigger output.

Send the following command for the digital output line which is to be used for trigger output (<TrigOutID>):

CTO <TrigOutID> 7 *polaritycode*

Example: The signal polarity for the digital output line 1 is to be set to "active low". Send:

CTO 1 7 0

3.10 Updates

3.10.1 Software Updates

Updated releases of software and manuals are available for download at www.pi.ws. You need a user name and a password to download software and manuals. These login data are provided on the Mercury product CD in the Releasenews PDF file in the \Manuals directory.
To download the latest software (complete CD mirror) from the PI Website, proceed as follows:

1. On the www.pi.ws home page, move the cursor to *Manuals, Software, ISO Statements* in the *Service* section on the left.
2. Select *Software* from the list that pops up.
3. On the *PI Support Site* page, enter the user name and the password which are provided in the C-663 Releasenews xxxxx.pdf on the Mercury product CD and click *Login.*
4. Click category *C* (Motion Controllers).
5. Click *C-663* > *C-663.11* > *Software* (if you click *Documents*, you will get the latest manuals).
6. Click the latest CD mirror (includes the manual versions that were with the release) or the latest update zip file.

3.10.2 Updating PIStages2.dat

To install the latest version of *PIStages2.dat* from the PI Website proceed as follows:

1. On the www.pi.ws home page, move the cursor to *Manuals, Software, ISO Statements* in the *Service* section on the left.
2. Select *Software* from the list that pops up.
3. On the *PI Support Site* page, click the *General Software* category (no login or password is required).
4. Click *PI Stages.*
5. Click *pistages2.*
6. In the download window, switch to the ...\PI\GcsTranslator directory. The location of the PI directory is that specified upon installation, usually in C:\Documents and Settings\All

Users\Application Data (Windows XP) or C:\ProgramData (Windows Vista and Windows 7) (may differ in other language versions of Windows).

Note that in PIMikroMove, you can use the *Version Info* entry in the controller menu or the *Search for controller software* entry in the *Connections* menu to identify the GcsTranslator path.

7 If desired, rename the existing PIStages2.dat (if present) so as to preserve a copy for safety reasons.

8 Download the file from the server as PIStages2.dat.

3.10.3 Firmware Updates

The current firmware revision of your C-663 is contained in the response to the *IDN? command.

- Firmware updates can be made by running the TMS320F28XX Updater firmware update program on the host computer. The program is available on request from our customer service department (see p. 184).

NOTES

The C-663 whose firmware is to be updated must be directly connected to the host PC (no daisy chain, do not even connect a cable to "RS-232 Out"), and the connection should be made via the RS-232 interface. USB connections are not recommended for firmware updates.

If the controller is in firmware update mode, the DIP switch settings for baud rate and controller address are ignored. The serial connection to the host PC is made with an automatic baud rate setting ("Autobaudrate") in the firmware update program. If the Autobaudrate connection should fail, try again to establish the connection.

When the controller is in firmware update mode, *all* LEDs on the front panel will stay off.

Proceed as follows to update the C-663 firmware:

1. Only required if the firmware update introduces new parameters (see the documentation that comes with the update):
   In the PITerminal or the *Command Entry* window of PIMikroMove, send SPA? and save the response to a text file for later restoration of the controller parameter values.

2. Power down the C-663.
   Set DIP switch 8 on the C-663 front panel to the the ON position (firmware update mode).

3. Start the TMS320F28XX Updater firmware update program.

4. Power on the C-663.

5. Establish communication between C-663 and host PC in the firmware update program (Autobaudrate connection).

6 Perform the update:
a) Select the new bootloader file and the new flash file for the update.
b) Make sure that the correct files are selected in the corresponding fields.
c) Start the update process by clicking the *Mercury Update* button.

If an error message is displayed saying that you must restart your controller, switch your controller off and on again. Wait at least 10 seconds, then click the *Mercury Update* button again.

7 When the update has finished, close the firmware update program.

8 Power down the C-663.
Set DIP switch 8 back to the OFF position (normal operation).

9 Power on the C-663. If the firmware update has not introduced new parameters, the C-663 can be started for normal operation with the new firmware. Otherwise proceed with step 10.

10 Only required if the firmware update introduces new parameters (see the documentation that comes with the update):
Make sure you have created a parameter backup file (see step 1).
In the PITerminal or the *Command Entry* window of PIMikroMove, send
ZZZ 100 Parameter
to set the new parameters to initial values. Since this command also resets all other parameters, you have to set them back to the values stored in the backup file using SPA (see "Controller Parameters" (p. 27) for details on parameter handling and saving). Furthermore, check the new parameters with SPA? and set them to plausible values.

4 System Description

4.1 Basic Elements

For successful operation of the C-663, you should familiarize yourself with the following features of the device.

Logical Axes:
The C-663 controls one logical axis of a mechanical system. See "Accessible Items and Their Identifiers" (p. 51) for details.

Input and Output Signals:
Input and output signals can be used for triggering purposes, and the input signals of a joystick can furthermore be used for velocity control of the C-663. See "Accessible Items and Their Identifiers" (p. 51) for details.

Communication Interfaces:
The C-663 can be controlled from a host computer (not included) with ASCII commands sent via:

- RS-232 serial connection
- USB interface: The USB drivers will make the USB interface appear to all software on the host PC as a new COM port. That port will be present only when the controller is connected via USB and powered on.

!

CAUTION

Never connect the RS-232-IN and USB connectors of the same controller to a PC at the same time as this can cause damage to the controller.

Up to 16 C-663 controllers can be controlled from a single host computer interface via a daisy chain.

See "Connecting Controller or Daisy-Chain Network to Host PC" (p. 25) for more information.

Controller Firmware:

The firmware comprises the ASCII command set and the controller parameters. For version information and updates see "Firmware Updates" (p. 47).

- ASCII Commands:

  The C-663 understands the PI General Command Set (GCS; version 2.0).
  The PI General Command Set (GCS) is supported by a wide range of PI systems. This command set is well-suited for positioning tasks with one or more axes. The command set itself is independent of the specific hardware (controller or attached stages).

  Commands are used, for example, to set operating modes, to initiate motion of the mechanical system and to query system and motion values. See "GCS Commands" (p. 72) for more information.

- Controller Parameters:

  The hardware basics of the connected stage and the required control settings are mirrored in controller parameters. Parameters can be modified by the user to adapt the system to the individual application. See "Controller Parameters" (p. 27) and "Customizing the System" (p. 65) for more information.

- Special Features:

  Data recorder: The C-663 comprises a real-time data recorder. It is able to record several signals (e.g. current position, analog input) from different data sources (e.g. logical axes or input channels). See "Data Recording" (p. 64) for more information.

  Macros: The C-663 can store macros. The macro feature allows defining command sequences and storing them permanently in the non-volatile memory of the device. It is possible to define a macro that will be executed automatically every time the C-663 is started, facilitating stand-alone operation without a host computer. See "Working with Controller Macros" (p. 59) for more information.

Software on Host PC

Usually, a host computer is used to operate or at least configure the C-663. Therefore, a wide range of software tools for installation on the host computer comes with the C-663. For a complete list of all software on the Mercury product CD, see "Software Description" (p. 10).

4.2 Accessible Items and Their Identifiers

The identifiers listed below are used to address the appropriate items with the commands of the PI General Command Set (GCS) which is supported by the firmware of the C-663:

- Logical axis: one axis, the default identifier is 1.
  In the C-663 firmware, motion is commanded for logical axes (i.e. for the directions of motion of a stage; move commands: e.g. MOV (p. 130) and MVR (p. 131)).
  The axis identifier can be changed using the SAI command (p. 138). It can consist of up to 8 characters; valid characters are 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ-_
  (ask with the TVI? command (p. 155)). The new axis identifier is saved automatically and thus still available after reboot or next power-on. You can ask with SAI? (p. 138) for the current valid axis identifier.

- Analog input channels: six channels, the identifiers are 1 to 6 (cannot be changed).
  "Genuine" analog input lines, with the identifiers 1 to 4, are Input 1 to Input 4 on the I/O socket (p. 191). Their number is reported by the TAC? command (p. 150), and their values can be queried with the TAV? command (p. 151). Note that these lines can also be used for digital input (see below).
  Further analog input lines are located on the Joystick socket (p. 193): channel 5 is the input line for the joystick axis, and channel 6 is the input line for the joystick button. They are not reported by by TAC? and TAV? commands. See also the Joystick information below.
  The values of all six channels can be recorded using the record option 81 of the DRC command (p. 98).

- **Digital output lines**: four lines, the identifiers are 1 to 4 (cannot be changed).
  1 to 4 identify the Output 1 to Output 4 digital output lines on the I/O socket (p. 191).
  See "Using Trigger Input and Output" (p. 39) for more information.

- **Digital input lines**: four lines, the identifiers are 1 to 4 (cannot be changed).
  1 to 4 identify the Input 1 to Input 4 digital input lines on the I/O socket (p. 191) which can also be used for analog input (see above).
  See "Using Trigger Input and Output" (p. 39) for more information.

- **Joystick**: one joystick device, identifier is 1 with all joystick-related commands (cannot be changed). Note that the second joystick device shown in some responses is an analog input which is currently deactivated and provided for future applications.
  The C-663 supports one axis and one button of the joystick.

  The input values of the joystick axis and the joystick button can be recorded using the record option 81 of the DRC command.

  The identifier of the joystick axis is 1 with joystick-related commands (JAS? (p. 114), JAX (p. 115), JAX? (p. 115), JDT (p. 116), JLT (p. 117), JLT? (p. 118)) and 5 with the DRC command, record option 81.
  The identifier of the joystick button is 1 with the joystick-related JBS? command (p. 116) and 6 with the DRC command, record option 81.
  See "Joystick Control" (p. 56) and "Joystick Socket" (p. 193) for more information.

- **Data recorder tables** (memory tables for recorded data): 2 tables with 1024 points per table, the identifiers are 1 and 2 (cannot be changed).
  See "Data Recording" (p. 64) for more information.

- **Controller address**: the C-663 device address in the range of 1 to 16 can be set with the DIP switches on the front panel, see "DIP Switch Settings" (p. 21) and "Target and Sender Address" (p. 75) for details. Each C-663 must have a unique controller address. It might therefore be required to change the controller address if a daisy chain network is to be set up.

4.2.1 Trajectory Generation

A trajectory generator performs calculations to determine the current position, velocity and acceleration of the axis at any given moment in time ("motion profile"). These values are called the commanded values (see also "Control Value Generation" (p. 53)). The profile that is created by the trajectory generator of the C-663 depends on the motion parameters given by corresponding commands, by controller parameters and/or by a joystick:

Motion Parameter	Corresponding Commands	Corresponding Controller Parameter	Notes
Acceleration (A)	ACC (p. 85), ACC? (p. 85)	Current acceleration (parameter ID 0xB; user unit/s^2) Changed by ACC command (p. 85) or by SPA / SEP, can be saved with WPA	Limited by parameter 0x4A (Maximum acceleration)
Deceleration (D)	DEC (p. 95), DEC? (p. 95)	Current deceleration (parameter ID 0xC; user unit/s^2) Changed by DEC command (p. 95) or by SPA / SEP, can be saved with WPA	Limited by parameter 0x4B (Maximum deceleration)
Velocity (V)	VEL (p. 158), VEL? (p. 158)	Current velocity (parameter ID 0x49; user unit/s) Changed by VEL command (p. 158) or by SPA / SEP, can be saved with WPA	Limited by parameter 0xA (Maximum velocity) A joystick connected to the C-663 which is enabled with the JON command (p. 120) applies a factor to the current velocity set with VEL, see "Joystick Control" (p. 56) for details.
Target Position	MOV (p. 130), MVR (p. 131), GOH (p. 110), STE (p. 147)	-	A joystick connected to the C-663 which is enabled with the JON command (p. 120) sets the travel range limits as target position. When disabling a joystick, the target position is set to the current position for joystick-controlled axes. See "Joystick Control" (p. 56) for details. When the motor is switched on with the SVO command (p. 149) or when axis motion has been stopped with #24 (p. 84), STP (p. 148) or HLT (p. 112), the target position is set to the current position.

The trajectory generator of the C-663 supports trapezoidal point-to-point profiles only: The axis accelerates linearly (at the given acceleration value) until it reaches the given velocity. It continues in motion at that velocity, then decelerates linearly (using the deceleration value) until it stops at the specified target position.

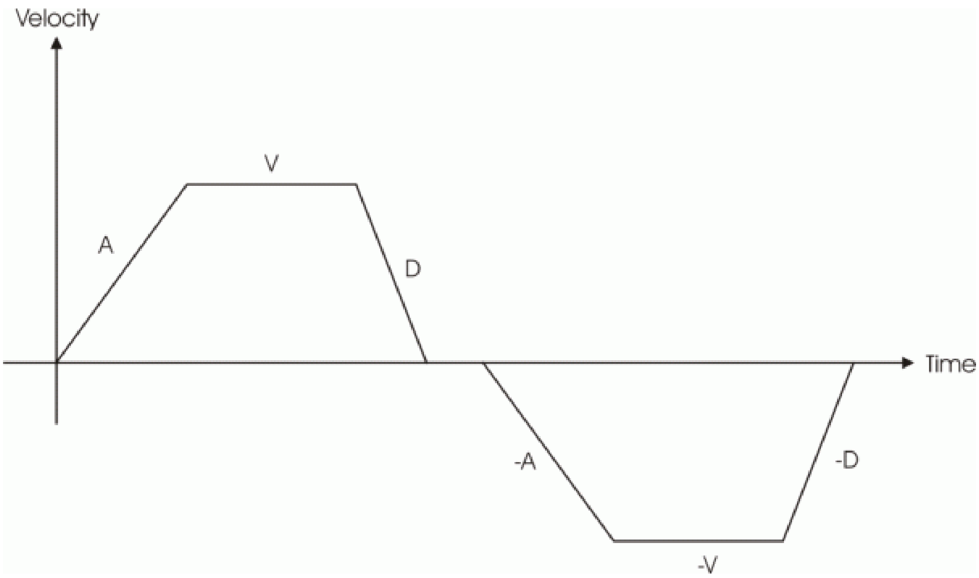


Figure 4: Simple trapezoidal point-to-point profiles, A = acceleration, D = deceleration, V = velocity

If deceleration must begin before the axis reaches the given velocity, the profile will have no constant velocity portion, and the trapezoid becomes a triangle.

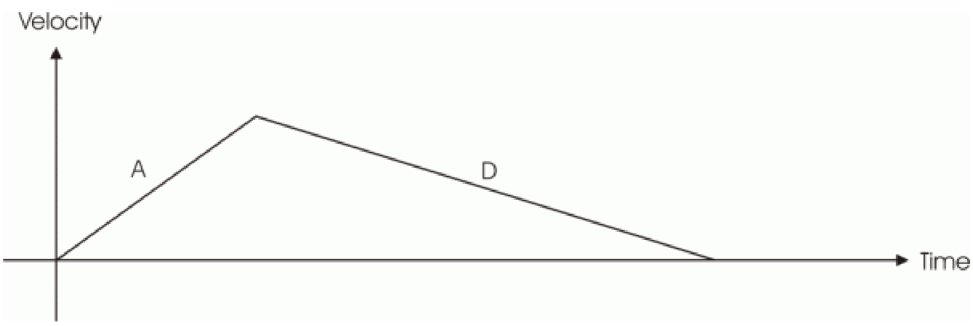


Figure 5: Simple trapezoidal point-to-point profile, A = acceleration, D = deceleration, no constant velocity

The slopes of the acceleration and deceleration segments may be symmetric (if acceleration equals deceleration) or asymmetric (if acceleration is not equal to deceleration). The acceleration parameter is always used at the start of the motion. Thereafter, the acceleration value will be used when the absolute velocity is increasing, and deceleration will be used when the absolute velocity is decreasing. If no motion parameters are changed during the motion then the acceleration value will be used until the maximum velocity is reached, and the deceleration value will be used when ramping down to zero.

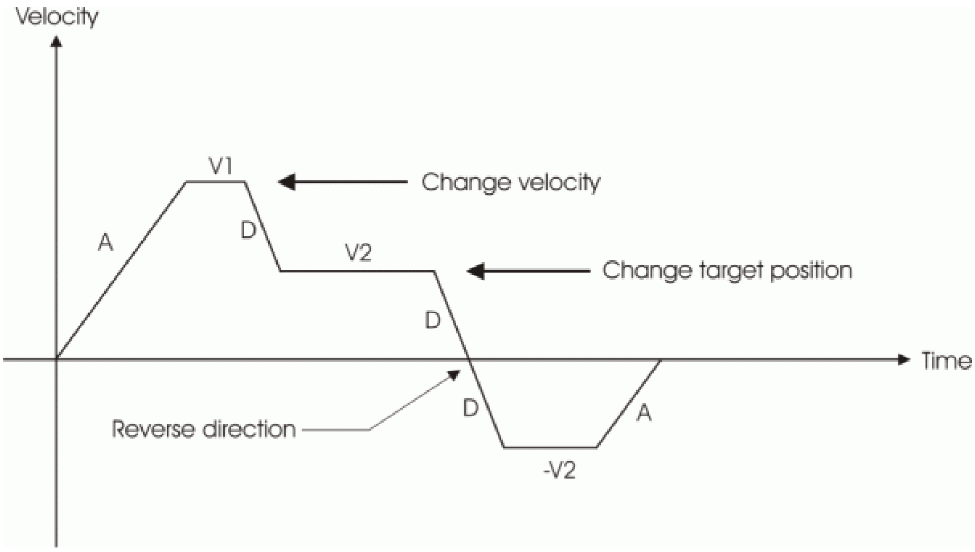


Figure 6: Complex trapezoidal profile, showing parameter changes; A = acceleration; D = deceleration; V1, V2, -V2 = velocities

It is acceptable to change any of the motion parameters while the axis is moving. The profile generator will always attempt to remain within the legal bounds of motion specified by the parameters. If, during the motion, the target position is changed in such a way that an overshoot is unavoidable, the profile generator will decelerate until stopped, then reverse direction to move to the specified position.

5 Joystick Control

CAUTION

Do not enable a joystick via command when no joystick device is connected to the controller hardware. Otherwise the corresponding controller axis may start moving and could damage your application setup.

!

C-663 controllers offer convenient manual motion control. One C-819.20 analog joystick device can be connected to the Joystick socket (p. 193) of the C-663. The C-663 supports one axis and one button of the joystick device. See "Accessible Items and Their Identifiers" (p. 51) for the joystick-related identifiers to use in commands.

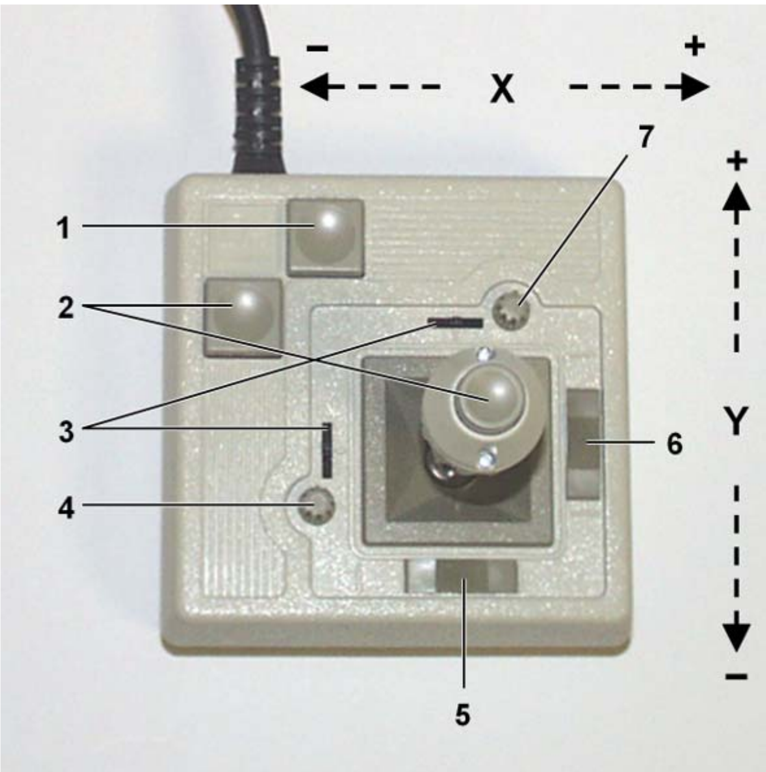


Figure 7: C-819.20 joystick device

1	Push button #1
2	Push button #2
3	Adjustment indicators
4	Y adjust
5	Spring release X
6	Spring release Y
7	X adjust

The two joystick axes of a C-819.20 joystick device can be connected through the C-819.20Y cable to two C-663 controllers. In this case, on both controllers the joystick axis 1 must be selected for operation (the Y-cable maps the signals internally). If the "Axis Y" branch of the Y-cable is to be used, the "Axis X" branch must also be connected to a controller to provide the supply power for the joystick device.

When a joystick device is connected directly to the controller and enabled, it is the velocity of the motion axes ("commanded velocity" output by the trajectory generator) that is determined by the displacement of the corresponding joystick axes. A lookup table defines the velocity response at a certain amplitude of the joystick axis. The values in a lookup table are factors which will during joystick control be applied to the velocity set with VEL (p. 158) for the controller axis, the range is -1.0 to 1.0. To change a lookup table, you can load profiles provided by the controller (power-on default = linear profile), or you can write a custom profile to the lookup table point-by-point (256 values).

During joystick control, the target position is set to the travel range limits given by the Max_Travel_Range_pos and Max_Travel_Range_neg parameters (0x15 and 0x30, see "Travel Range Adjustment" (p. 68) for more information). For joystick operation, the motor of the stage must be switched on with SVO. When disabling a joystick, the target position is set to the current position for joystick-controlled axes.

NOTE

Before a joystick can be operated correctly, a calibration routine may need to be performed. Activating the joystick before calibration may cause the axis to start moving even though the joystick is in the neutral position. To calibrate a joystick axis of the C-819.20, turn the corresponding "Adjust" knob on the joystick until the axis stops.

Commands for joystick handling:

JON (p. 120)	Enables or disables a specified joystick device for joystick operation. While a joystick is active on a controller axis, move commands are neither accepted from the command line nor from macros for that axis.
JON? (p. 121)	Queries the current activation state of the joystick devices.
JAX (p. 115)	Sets the controller axis which is to be controlled by a joystick axis. Each axis of a controller can only be controlled by one joystick axis.
JAX? (p. 115)	Queries the current assignment of controller axes to joystick axes.

JAS? (p. 114)	Queries the current status of joystick axes. The response corresponds to the current displacement of the joystick axis and is the factor which is currently applied to the current valid velocity setting of the controlled motion axis, according to the lookup table.
JBS? (p. 116)	Queries the current status of joystick buttons. The response indicates if the joystick button is pressed; 0 = not pressed, 1 = pressed.
JDT (p. 116)	Sets predefined lookup table types for the joystick axes. The current lookup table content for the specified joystick axis is overwritten by the selection made with JDT. The C-663 provides the following lookup table types for special applications: 1 = linear (default) 2 = parabolic
JLT (p. 117)	Fills the lookup table for a joystick axis point-by-point. This overwrites the current lookup table content for this joystick axis. See the command description for an example.
JLT? (p. 118)	Reads the current lookup table values.

6 Working with Controller Macros

The macro feature allows defining command sequences and storing them permanently in the non-volatile memory of the controller. Each defined macro can be called up by its own user-defined name. In addition, it is possible to define a macro that will be executed automatically every time the C-663 is started, making possible stand-alone operation without a host computer. See the subsections below and the MAC command (p. 124) description for more details and examples.

NOTE

PIMikroMove offers a comfortable macro editor on the *Controller macros* tab. Furthermore, PIMikroMove offers the "Host macro" feature which makes it possible to save macros on the host PC.

6.1 Defining Macros

To define a macro command sequence, first activate macro recording mode with the command MAC BEG <macroname> where <macroname> is a name of your choice with a maximum of 8 characters. While in macro recording mode, commands are not executed but stored in macro storage. Recording mode is exited by the MAC END command.

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

During macro recording no macro execution is allowed.

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

A running macro sends no responses to any interface. This means questioning commands are allowed in macros but not answered and therefore useless.

In macros you can set local variables using the VAR (p. 156), ADD (p. 86) and CPY (p. 89) commands. See "Variables" (p. 76) for more information.

The following commands provided by the C-663 can only be used in macros:
DEL (p. 96), MEX (p. 128), WAC (p. 159) and JRC (p. 122). Using MEX and WAC, it is possible to set stop conditions or conditions for further macro processing. JRC will jump to a specific line within a macro depending on a given condition.

Macro recording is possible when a joystick is active on the axis.

Example 1: Note how macro3 calls macros #1 and #2 for execution.

```
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
```

During macro recording for a controller whose address is different from 1, the address must as target ID be part of each command line to be recorded, but will not become part of the macro content.

Example 2: The controller address is set to 2 with the DIP switches. Macro addrtest is to be recorded:

```
2 MAC BEG addrtest
2 SVO 1 1
2 DEL 1000
2 FRF 1
2 MAC END
```

Now you can check the content of macro addrtest by sending

```
2 MAC? addrtest
```

The answer is

```
SVO 1 1
DEL 1000
FRF 1
```

i.e. the target ID has not become part of the macro.

See "Target and Sender Address" (p. 75) for more information.

6.2 Starting Macro Execution

A defined macro can be run by the command
MAC START <macroname> [<String1> [<String2>]]
where <macroname> is the name that was given to the macro to be run.

To run a macro multiple times, call it with
MAC NSTART <macroname> <uint> [<String1> [<String2>]]
where <uint> gives the number of times the macro is to be run.

<STRING1> and <STRING2> are optional arguments which give the values for the local variables 1 and 2 used in the specified macro. <STRING1> and <STRING2> can be given directly or via the values of variables. See the MAC command description (p. 124) and "Variables" (p. 76) for more information.

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

Any commands can be sent from the command line when a macro is running. The macro content and move commands received from the command line may overwrite each other, and only the last move command will be executed, irrespective of its source.

Macro execution can be stopped from the command line with #24 (p. 84), STP (p. 148) and HLT (p. 112).

A running macro may not be deleted.

When an error is caused by the running macro, you can use parameter 0x72 (Ignore Macro Error, p. 35) to define whether the macro should stop or continue to run.

Macro execution is not allowed when a joystick is active on the axis. See "Joystick Control" (p. 56) for details.

You can query with #8 (p. 83) if a macro is currently running on the controller and ask for the names of currently running macros with the RMC? command (p. 135).

6.3 Start-Up Macro

With MAC DEF <macroname> it is possible to set the specified macro as start-up macro. This macro will be automatically executed with the next power-on or reboot of the controller.

Example:

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

In the example, axis 1 will after power-on be immediately ready for operation since the start-up macro switches the motor on and performs a reference move to the negative limit switch.

To ask for the current start-up macro setting, send

```
MAC DEF?
```

To undo the current start-up macro selection, send

```
MAC DEF
```

i.e. omit <macroname>.

Deleting a macro with MAC DEL <macroname> does not delete the start-up macro selection.

6.4 Example: Synchronization of Two Controllers

Two C-663 controllers can execute synchronized moves using a connecting cable and two macros running on the master and slave controller.

Hardware requirements: Connect Output 1 line of the master controller's I/O socket to Input 1 line of the slave controller's I/O socket. See "I/O Socket" (p. 191) for the lines and pinout.

How to proceed:

1 Prepare the motion on both controllers by sending a sequence like:

```
VEL 1 0
MOV 1 5.5
```

2 Define macros for the master controller and for the slave controller which simply set the velocity to a value different from 0:

The master controller's macro is as follows:

```
MAC BEG master
DIO 1 1
VEL 1 100
MAC END
```

The slave controller's macro is as follows:

```
MAC BEG slave
WAC DIO? 1 = 1
VEL 1 100
MAC END
```

3 Start the macro on the slave controller.

4 Start the macro on the master controller. Both axes will start moving as their velocities are different from 0 (for the slave axis, the velocity setting is triggered by Output line 1 of the master controller).

7 Data Recording

For general information regarding the data recording you can send the HDR? command (p. 111) which lists available record options and trigger options and gives additional information about data recording. The C-663 has 2 data recorder tables (ask with TNR? (p. 153)) with 1024 data points per table.

The data recorder configuration, i.e. the assignment of data sources and record options to the recorder tables, can be changed with DRC (p. 98), and the current configuration can be read with DRC? (p. 99). Data recorder tables with record option 0 are deactivated, i.e. nothing is recorded.

Recording can be triggered in several ways. Ask with DRT? (p. 102) for the current trigger option and use DRT (p. 101) to change it. A trigger option set with DRT will become valid for all data recorder tables with non-zero record option. By default data recording is triggered when a step response measurement is made with STE (p. 147).

The record table rate can be set with the RTR command (p. 137). The power-on default of this value is 10 (the unit is controller cycles; ask with RTR? (p. 138)). You can cover longer periods by increasing the record table rate. Note that the cycle time of the C-663 is 50 µs.

Recording always takes place for all data recorder tables with non-zero record options.

Recording ends when the content of the data recorder tables has reached the maximum number of points.

The last recorded data can be read with the DRR? command (p. 99). The data are reported in GCS array format, for details regarding the GCS array see the separate manual (SM146E) which is provided on the Mercury product CD. Reading can take long depending on the number of points to be read! It is possible to read the data while recording is still in progress.

When the controller is powered down, the content of the data recorder tables and all data recorder configuration and trigger settings are lost. The configuration and trigger settings are reset to their factory defaults on power on.

8 Customizing the System

The procedure described in the "Adding Stages to User DAT Files" (p. 65) subsection normally is required in the following cases:

- You integrate drives in a custom system → see also the user manual of the motor for details.
- Your application requires a moving mass > 20 g, vertical motion or presence of external force, or highest performance.

When you replace your stage with one of another stage type, see "Parameter Databases" (p. 37) and "Adding Stages to User DAT Files" (p. 65) for more information.

If your application requires a reduced travel range and/or a modified home position, read "Travel Range Adjustment" (p. 68).

You can use PIMikroMove to store parameter values as default settings to the controller's non-volatile memory. See "Changing Default Parameter Values" (p. 66) for instructions.

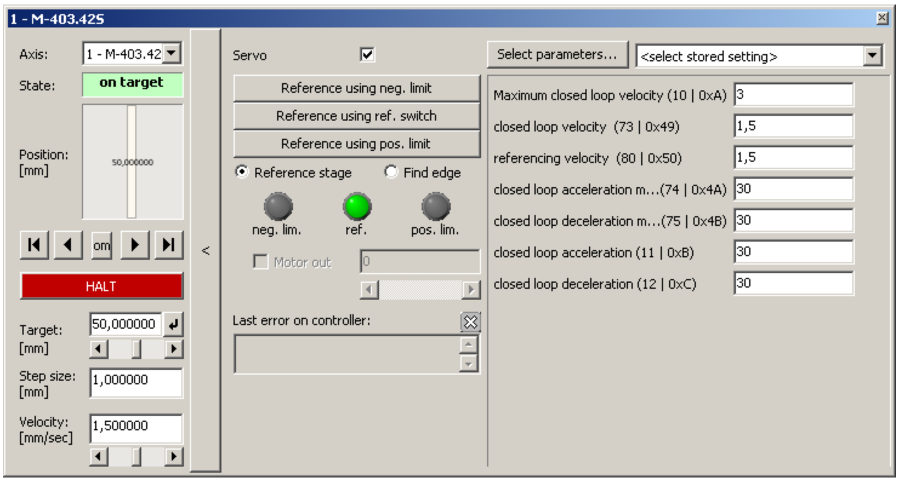
8.1 Adding Stages to User DAT Files

Whenever you design a new mechanical system or new application requirements like changes in load or force arise, the controller parameters describing the stage must be adapted accordingly. To make the new parameter values available in the UserStages2 database used by the host software from PI, you have to add a new parameter set (= stage).

The easiest way to do this is to modify the parameters of an existing stage type and save them under a new name. Thereafter you can select this newly defined stage in PIMikroMove or in other PI software as well.

Proceed as follows to add a new stage to your UserStages2 database (e.g. PI_UserStages2.dat):

1. In PIMikroMove, assign the stage type that comes closest to your stage to the appropriate axis. See "First Steps" (p. 15, step 8) for details. Afterwards the *Start Up Axes* dialog may open—you can close this dialog because at this point it is not necessary to reference the axis.
2. Open the Single-Axis Window for the axis (see the PIMikroMove manual for more information). To do this, go to *View > Single Axis Window* and select your stage.
3. Expand the Single-Axis Window via the rightmost > button.



4. In the rightmost pane of the expanded Single-Axis Window, display the columns for the parameters you want to modify. To do this, click the *Select parameters…* button (see the PIMikroMove manual for more information).
5. Type new values in the parameter fields. As long as a value is shown in blue, it is only present in PIMikroMove but not yet sent to the controller. Press Enter on your keyboard to send the value to the controller's volatile memory.
6. Right-click in the center pane of the expanded Single Axis Window and select *Add/Edit User Stage type...* from the menu that appears.
7. To save the modified settings as a new stage, enter a new name for your stage in the *as stage type* field and click *OK*.

   For the stage type entry, do not use stage names which already exist in the PIStages2.dat database. If a stage of the same name exists in PIStages2.dat and UserStages2.dat, the parameter settings from PIStages2.dat will be preferred when assigning that stage to an axis (e.g. in the *Select connected stages* dialog; see p. 15, step 8), and the settings from UserStages2.dat will never be used.

The new stage is now displayed on the *Axes* tab, and you can work with it (e.g. reference the stage: right-click the axis row and select the *Start up axes...* item). If you want to further modify the stage parameters, use the *Add/Edit User Stage type* menu item again to save the changes.

8.2 Changing Default Parameter Values

You can customize the default settings of your controller by saving the stage parameters to the controller's non-volatile memory. Before doing this, make sure that you have the correct parameter settings.

To customize and save the parameters, proceed as follows:

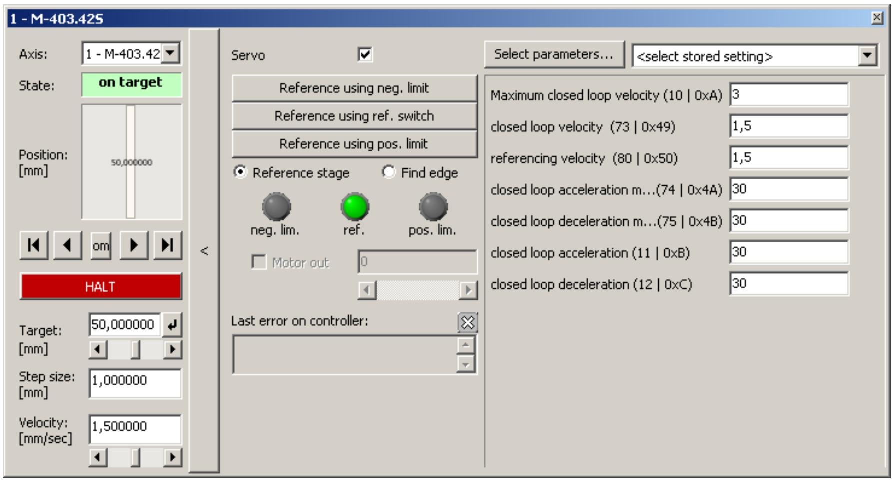
1 Start PIMikroMove.

2 Select a suitable stage in PIMikroMove (see "First Steps", p. 15, step 8).

3 Customize your parameters as needed:

3.1 Open the *Single Axis Window* for the connected stage by selecting *View > Single Axis Window* from the menu.

3.2 Expand the *Single Axis Window* via the rightmost > button.



3.3 In the list on the right side of the expanded *Single Axis Window*, check the current parameter settings and enter new parameter values.
If a parameter you wish to adjust is not listed, click *Select parameters…* to display the parameter in the list.

If you enter new parameter values, they will be highlighted in blue. Note that you must press <ENTER> on your keyboard to send them to the controller's volatile memory. Once the new parameter values have been sent, they will no longer be highlighted in blue.

4 Choose *Save parameters to non-volatile memory* from the controller menu (e.g. *C-663 (COM1) > Save parameters…)*.
This opens a new window.

5 Enter the password "100" for the WPA command and click *OK*.
The current valid values of the controller parameters are saved to the non-volatile memory where they become the default values.

8.3 Travel Range Adjustment

The figures below give a universal hardware scheme of a positioning stage with incremental sensor, reference and limit switches. To work with such a stage, the corresponding controller parameters must be adjusted properly (see "Controller Parameters" (p. 27) for how to modify parameter values).

In the example shown in the first figure, the travel range, i.e. the distance from negative to positive limit switch is 20 mm, the distance between the negative limit switch and the reference switch is 8 mm, and the distance between reference switch and positive limit switch is 12 mm (you can use the FED command (p. 103) in combination with the POS? command (p. 134) to identify the values). These hardware properties are represented by the following controller parameters in the C-663 firmware:

DISTANCE_REF_TO_N_LIM (parameter ID 0x17) = 8
DISTANCE_REF_TO_P_LIM (parameter ID 0x2F) = 12

To allow for flexible localization of the home position (0), a special parameter is provided. It gives the offset between reference switch and home position which is to be valid for the stage after a reference move (see below). In the example, the home position is to be located at the negative limit switch after a reference move, and hence the offset between reference switch and home position is 8 mm.

VALUE_AT_REF_POS (parameter ID 0x16) = 8

To allow for absolute moves, either an absolute "initial" position can be set with the POS command (p. 133), or the stage can perform a reference move to a known position where a defined position value will be set as the current position (see "Referencing" (p. 37) for further details). By default, a reference move is required. In the example, known positions for reference moves are given by the reference switch and the limit switches. Depending on the switch used for the reference move, a certain combination of the above-mentioned parameters is used to calculate the position to be set at the end of the move:

- Reference switch (FRF command (p. 108)): the stage is moved to the reference signal, and the value of VALUE_AT_REF_POS is set as the current position.

- Negative limit switch (FNL command (p. 105)): the stage is moved to the negative limit signal and the difference of VALUE_AT_REF_POS and DISTANCE_REF_TO_N_LIM is set as the current position (can be negative).

- Positive limit switch (FPL command (p. 107)): the stage is moved to the positive limit signal and the sum of VALUE_AT_REF_POS and DISTANCE_REF_TO_P_LIM is set as the current position.

It is furthermore possible to set "soft limits" which establish a "safety distance" which the stage will not enter on both ends of the travel range. In the C-663 firmware, those soft limits always refer to the current home position (0; in the example located at the negative limit switch after a reference move). The soft limits are to be deactivated in the example so that the corresponding parameters must be as follows:

MAX_TRAVEL_RANGE_POS (parameter ID 0x15) = 20 mm
MAX_TRAVEL_RANGE_NEG (parameter ID 0x30) = 0 mm
(This means that the stage can move 20 mm in positive direction, starting from the home position, and 0 mm in negative direction, starting from the home position.)

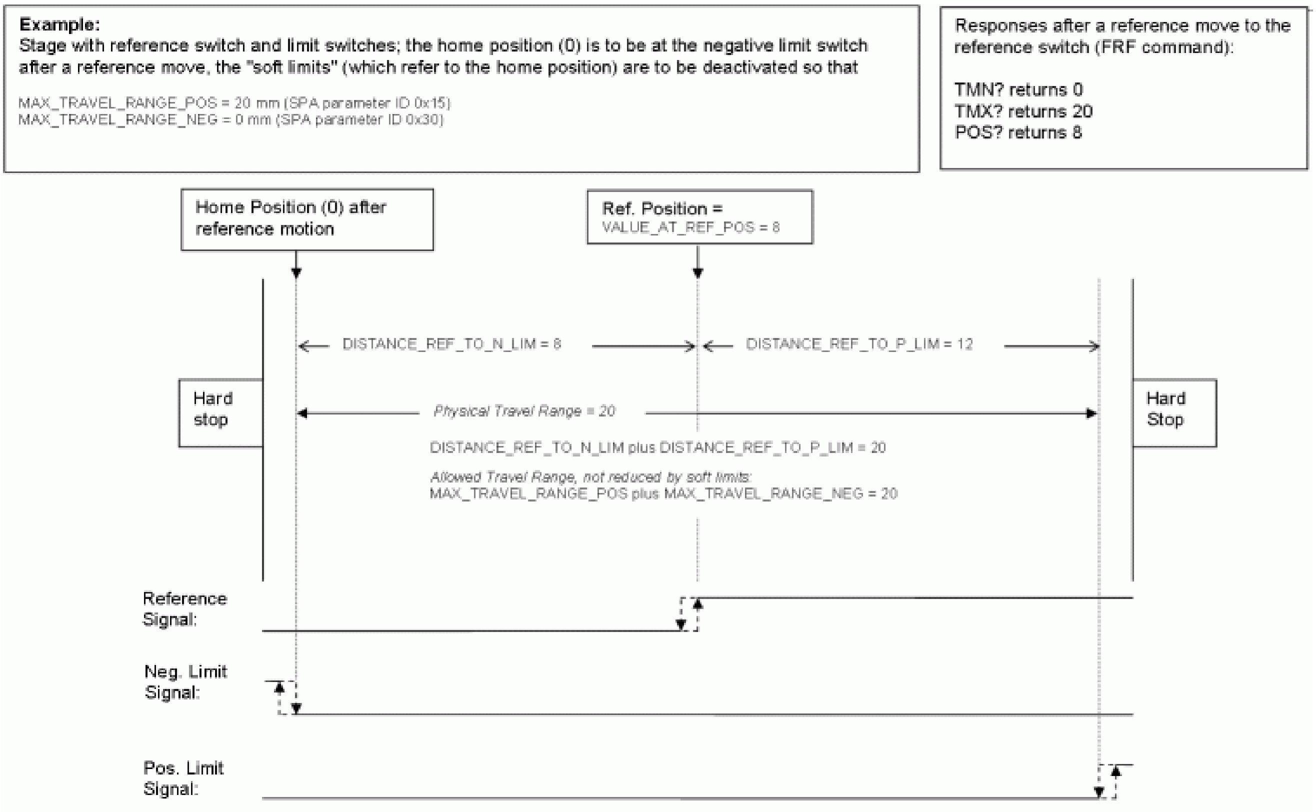


Figure 8: Positioning stage and corresponding controller parameters

Now in the same example, a "safety distance" is to be established on both ends of the travel range by setting soft limits, and the home position is to be located at about 1/3 of the distance between the new negative end of the travel range and the reference switch. The limit switches cannot be used for reference moves anymore.

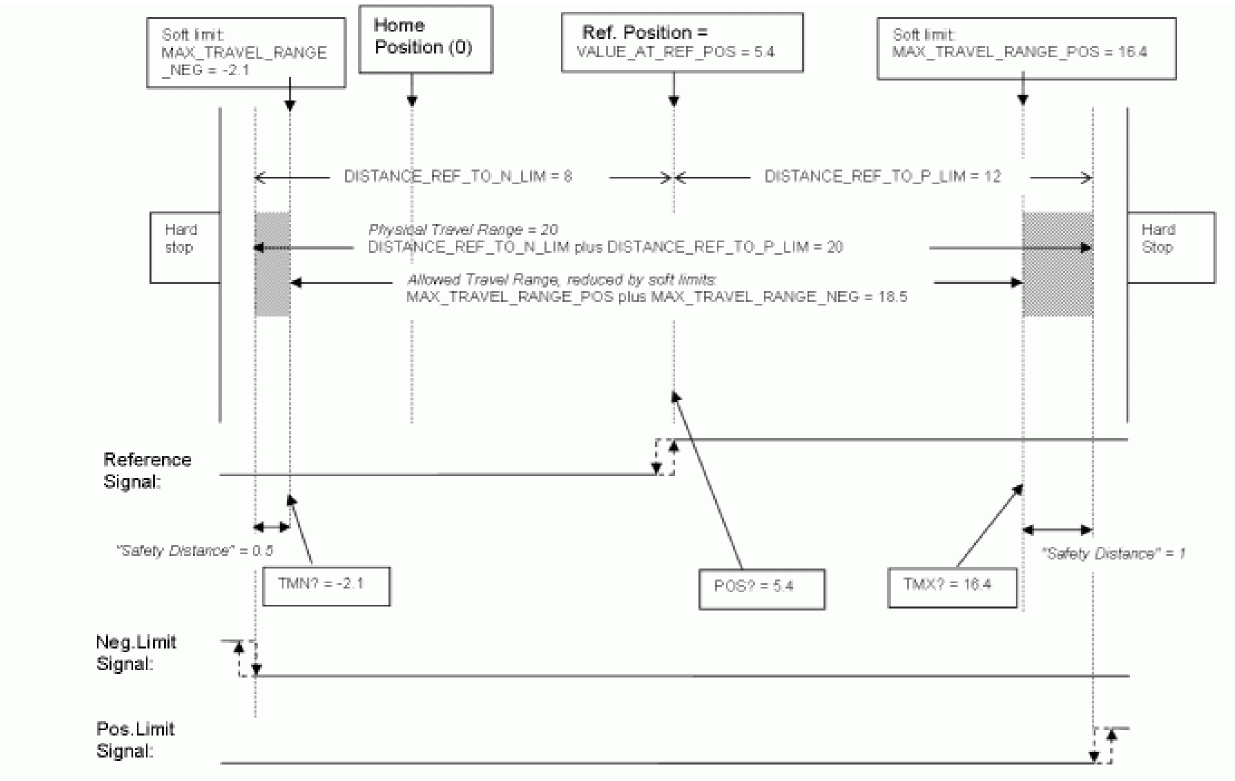


Figure 9: Positioning stage, soft limits set in the controller to reduce the travel range

After the stage was referenced again by moving it to the reference switch (FRF command), the following responses will be given:

TMN? (p. 152) returns -2.1
TMX? (p. 152) returns 16.4
POS? (p. 134) returns 5.4

!

CAUTION

If the soft limits (MAX_TRAVEL_RANGE_POS and MAX_TRAVEL_RANGE_NEG) 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 switch can be used for a reference move (FRF).

Be careful when setting the values for VALUE_AT_REF_POS, MAX_TRAVEL_RANGE_POS and MAX_TRAVEL_RANGE_NEG because there is no plausibility check.

The soft limits may not be outside of the physical travel range:
MAX_TRAVEL_RANGE_POS ≤ DISTANCE_REF_TO_P_LIM + VALUE_AT_REF_POS
MAX_TRAVEL_RANGE_NEG ≥ VALUE_AT_REF_POS - DISTANCE_REF_TO_N_LIM
Otherwise, reference moves to the limit switches would have incorrect results because the values of the soft limits would be set at the end of the referencing procedure.

Be careful when referencing the stage by setting an initial absolute position with POS since the values for MAX_TRAVEL_RANGE_POS and MAX_TRAVEL_RANGE_NEG are not adapted. In the worst case, the soft limits will now be outside of the physical travel range, and the stage will no longer be able to move since the move commands check the soft limit settings.

NOTE

You can use the digital input lines as source of the reference signal, the negative limit signal or the positive limit signal. See parameters 0x5C (p. 31), 0x5D (p. 32) and 0x5E (p. 33) for details.

The digital input line selected as source of the reference signal is used for referencing irrespective of the setting of parameter 0x14 (“Stage has a built-in reference switch”).

While the built-in limit switches of the stage are processed only if enabled with parameter 0x32 (“stage has limit switches”), the setting of parameter 0x32 has no influence on the usage of activated digital input lines.

9 GCS Commands

The PI General Command Set (GCS) is supported by a wide range of PI systems. This command set is well-suited for positioning tasks with one or more axes. The command set itself is independent of the specific hardware (controller or attached stages).

Commands are used to set operating modes, initiate axis motion and to query system and motion values. Because of the variety of functions and parameters, a sequence of commands must often be transferred in order to achieve a desired system action.

You can type commands, for example, in the *Command entry* window of PIMikroMove, or in the PITerminal.

9.1 Format

9.1.1 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 (p. 51) or a command-specific parameter.
[…]	Square brackets indicate an optional entry.
{…}	Braces indicate a repetition of entries, i.e. that it is possible to access more than one item (e.g. several axes) in one command line.
LF	LineFeed (ASCII char #10) is the default termination character.
SP	Space (ASCII char #32)

9.1.2 GCS Syntax

Except as listed below, a GCS command consists of 3 characters, e.g. CMD. The corresponding query command has a "?" appended, e.g. CMD?. Command mnemonic:

CMD ::= character1 character2 character3 [?]

Exceptions:

- Special commands, e.g. fast polling commands, consist only of one character. The 24th ASCII character e.g. is called #24. Note that these commands are not followed by a termination character (but the responses to them are).
- *IDN? (for GPIB compatibility).

The command mnemonic is not case-sensitive.

General:

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

That means the command mnemonic and all arguments (e.g. axis IDs, channel IDs, parameters, etc.) must be separated from each other by one space.

Example:
Send: MOVSP1SP10.0LF
to move Axis 1 to position 10.0 (the unit depends on the controller, can be µm or mm, for example)

More than one command mnemonic per line is not allowed. Several groups of arguments following a command mnemonic are allowed, e.g.
MOVSP1SP17.3SP2SP2.05LF
if there were 2 axes. The command line ends with the termination character (LF).

If 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. For example,
RPALF
resets all parameters in volatile memory.

The <AxisID> argument is used for the logical axes of the controller. Depending on the controller, an axis could be identified with up to 16 characters – all alphanumeric characters and the underscore are allowed.

See "Accessible Items and Their Identifiers" (p. 51) for the identifiers supported by the C-663.

Definitions for query commands (report commands):

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

When all arguments are optional and are omitted, all possible values are reported. For example,
POS?
queries the position of all axes.

Reply syntax:

[<argument>[{SP<argument>}]"="]<value>LF

Multi-line reply syntax:
{[<argument>[{SP<argument>}]"="]<value>SP LF}
[<argument>[{SP<argument>}]"="]<value>LF for the last line!

The command
CMD?SP<arg3>SP <arg1>SP <arg2>LF
replies in the same order:
<arg3>"="<value3>SP LF
<arg1>"="<value1>SP LF
<arg2>"="<value2> LF

Example:
Send: TSP?SP2SP1
Report: 2=-1158.4405SP LF
1=+0000.0000LF

NOTE

With the C-663, you can address only one single item (e.g. axis or channel) per command line, or, if the command supports this, address all items by omitting the item identifier.

Example:

You can send

SEP 100 1 0x32 0

to save a new value of parameter 0x32 for axis 1 to non-volatile memory

but it is not possible to send

SEP 100 1 0x32 0 1 0x14 1

9.1.3 Target and Sender Address

In principle, the addresses of the target controller and the sender are required in every command line, even with single-character commands like #4 or when recording a macro. But because only the host PC may send command lines to the C-663s, its address (0) can be omitted. Both target and sender address, however, are part of any controller response.

Example:

In a terminal such as PITerminal, you can ask with the *IDN? command for the identification string of the C-663 with address 2 either by sending
2 0 *idn?
or by sending
2 *idn?
The response in either case is
0 2 (c)2011 Physik Instrumente(PI) Karlsruhe, C-663.11,0,1.2.0.1

Exception:

The target address can be omitted if the target C-663 has the address 1, even if this C-663 is part of a daisy chain. If the target address is omitted, the target and sender addresses will also be omitted in any response of the controller.

Example:

If you send
*idn?
the controller with the address 1 will respond with
(c)2011 Physik Instrumente(PI) Karlsruhe, C-663.11,0,1.2.0.1
If you send
1 *idn?
it will respond with
0 1 (c)2011 Physik Instrumente(PI) Karlsruhe, C-663.11,0,1.2.0.1

See "DIP Switch Settings" (p. 21) for how to set the controller address. Possible controller addresses are in the range of 1 to 16, address 1 is default. The host 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 reports are displayed on the host PC.

9.1.4 Variables

With the C-663, variables are provided for more flexibility in programming. 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 RAM only.

Conventions for 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 written with preceding "$".
- 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:

- Can only be used in macros
- At present, the controller firmware supports three local variables: 0, 1 and 2
- The values of the local variables 1 and 2 are given as arguments of the MAC START or MAC NSTART command when starting the macro. The command formats are:
  MAC START <macroname> [<String1> [<String2>]]
  MAC NSTART <macroname> <uint> [<String1> [<String2>]]
  <STRING1> and <STRING2> give the values for the local variables 1 and 2 used in the macro. <STRING1> and <STRING2> can be given directly or via the values of variables. <uint> gives the number of times the macro is to be run. See the MAC command description (p. 124) for more information.
  The local variable 0 is read-only. Its value gives the number of arguments set for the macro.
- Inside a macro, the values of local variables can be modified using ADD, CPY or VAR, 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:

- Can be used inside and outside of macros.
- Maximum number is 10
- Variables are created and modified using ADD, CPY or VAR. They can be deleted with VAR.
- The variable values can be queried with VAR?.

9.2 Command Survey

9.3 Command Reference (alphabetical)

#4 (Request Status Register)

Description: Requests system status information.

Format: #4 (single ASCII character number 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. 146), but only one character must be sent via the interface. Therefore #4 can also be used while the controller is performing time-consuming tasks.

For the C-663, the response is the sum of the following codes, in hexadecimal format:

Bit	15	14	13	12	11	10	9	8
Description	On Target	Is referen-cing	Is Moving	Motor On	-	-	-	Error flag

Bit	7	6	5	4	3	2	1	0
Description	Digital Input 4	Digital Input 3	Digital Input 2	Digital Input 1	-	Positive Limit	Reference	Negative Limit

Example: Send: #4
Receive: 0x9005
Note: The response is given in hexadecimal format. It means that the axis is on target, the motor is switched on, no error occurred, the states of the digital input lines 1 to 4 are low, and the stage is on the positive side of the reference switch (limits are not active, note that the logic of the signals is inverted in this example)

#5 (Request Motion Status)

Description: Requests motion status of the axes.

Format: #5 (single ASCII character number 5)

Arguments: none

Response: The answer <uint> is bit-mapped and returned as the hexadecimal sum of the following codes:

1=first axis is moving
2=second axis is moving
4=third axis is moving
...

Examples: 0 indicates motion of all axes complete
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).

Note: Use #5 (p. 83) instead of #7 to verify if motion has finished.

Format: #7 (single ASCII character number 7)

Arguments: none

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

B0h (ASCII character 176 = "°" in Windows) if controller is not ready
(e.g. performing a referencing command)

Troubleshooting: The response characters may appear differently in non-Western character sets or other operating systems. They may be indistinguishable on the controller screen.

#8 (Query If Macro Is Running)

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

Format: #8 (single ASCII character number 8)

Arguments: none

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

#24 (Stop All Axes)

Description:	Stops all axes abruptly. For details see the notes below. Sets error code to 10. This command is identical in function to STP (p. 148), but only one character must be sent via the interface. Therefore #24 can also be used while the controller is performing time-consuming tasks.
Format:	#24 (ASCII character 24)
Arguments:	none
Response:	none
Notes:	#24 stops all motion caused by move commands (e.g. MOV (p. 130), MVR (p. 131), GOH (p. 110), STE (p. 147)), referencing commands (FNL (p. 105), FPL (p. 107), FRF (p. 108)) and macros (MAC (p. 124)). After the axes are stopped, their target positions are set to their current positions. HLT (p. 112) in contrast to #24 stops motion with given system deceleration with regard to system inertia.

*IDN? (Get Device Identification)

Description:	Reports the device identity number.
Format:	*IDN?
Arguments:	none
Response:	One-line string terminated by line feed with controller name, serial number and firmware version
Notes:	For C-663, *IDN? replies something like: **(c)2011 Physik Instrumente(PI) Karlsruhe, C-663.11,0,1.2.0.1**

ACC (Set Acceleration)

Description: Set acceleration of given axes.

ACC can be changed while the axis is moving.

Format: ACC {<AxisID> <Acceleration>}

Arguments: <AxisID> is one axis of the controller

<Acceleration> is the acceleration value in physical units/s^2.

Response: none

Troubleshooting: Illegal axis identifiers

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

ACC changes the value of the Current acceleration parameter (ID 0xB) in volatile memory (can be saved as default with WPA (p. 160), can also be changed with SPA (p. 142) and SEP (p. 139)).

The maximum value which can be set with the ACC command is given by the Maximum acceleration parameter, ID 0x4A (can be changed with SPA (p. 142) and SEP (p. 139)).

ACC? (Get Acceleration)

Description: Get the current value of the acceleration.

If all arguments are omitted, gets current value of all axes.

Format: ACC? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller

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

where

<float> is the current active acceleration value in physical units / s^2.

ADD (Add and Save to Variable)

Description: Add two values and save the result to a variable.
Local variables can be set using ADD in macros only. See “Variables” (p. 76) for details regarding local and global variables.
The variable is present in RAM only.

Format: ADD <Variable> <FLOAT1> <FLOAT2>

Arguments:

<Variable> is the name of the variable to which the result is to be saved.

<FLOAT1> is the first summand.

<FLOAT2> is the second summand.
For the summands, floating point numbers are expected. They can be given directly or via the value of a variable.

See “Variables” (p. 76) for conventions regarding variable names and values.

Response: None

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

Value $A is subtracted from value $B, and the result is saved to variable C:
ADD C $B -$A

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

	Send:	ADD C $B -$A // subtract with "ADD"
	Send:	VAR? C
	Receive:	C=-345

Example 3:

Using the macros below, it is possible 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.

Variable value $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.
Variable value $2: Number of repetitions of the whole "flashing light" procedure.
DIO 0 <bitmask>: Sets the output channels according to <bitmask>. For example, "DIO 0 5" enables the channels 1 and 3 and disables all other channels (5 is 0000 0101 in binary numbers).

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

1. Write macro "STEPS":

```
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 macro "TEST":

```
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

BRA (Set Brake Activation State)

Description: Activates/deactivates brake for given axes.

Format: BRA {<AxisID> <BrakeState>}

Arguments: <AxisID> is one axis of the controller

<BrakeState> can have the following values:
0 = brake off
1 = brake on

Response: None

Troubleshooting: Illegal axis identifier

BRA? (Get Brake Activation State)

Description: Gets brake activation state of given axes.

If all arguments are omitted, gets status of all axes.

Format: BRA? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller

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

where

<BrakeState> is the current brake activation state of the axis:
0 = brake off
1 = brake on

Troubleshooting: Illegal axis identifier

CPY (Copy into Variable)

Description: Copy a command response into a variable.
Local variables can be set using CPY in macros only. See “Variables” (p. 76) for details regarding local and global variables.
The variable is present in RAM only.

Format: CPY <Variable> <CMD?>

Arguments:

<Variable> is the name of the variable to which the command response is to be copied. See “Variables” (p. 76) for name conventions.

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

Response: None

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 macro “connect”:
MAC BEG connect
CPY 1 DIO? 0
DIO 0 $1
MAC START CONNECT
MAC END

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

CPY TARGET VAR? SOURCE

CST? (Get Assignment of Stages to Axes)

Description:	Returns the name of the connected stage for the queried axis.
Format:	CST? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller
Response:	{<AxisID>"="<string> LF} where <string> is the name of the stage 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. 142) or SEP (p. 139). See also "Controller Parameters" (p. 27).

CSV? (Get Current Syntax Version)

Description:	Get current GCS syntax version used in the firmware.
Format:	CSV?
Arguments:	none
Response:	The current GCS syntax version (e.g. "2.0" for GCS 2.0)

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 <CTOPam> is the CTO parameter ID in decimal format, see below for the available IDs <Value> is the value to which the CTO parameter is set, see below
Response:	None

Notes:

The trigger output conditions will become active when activated with TRO (p. 153). Do not use DIO (p. 96) on output lines for which the trigger output is activated with TRO.

The CTO settings are lost when you power down or reboot the C-663. An easy way to keep them is to save them to a macro.

Available output lines and trigger conditions:

<TrigOutID> corresponds to the output lines Output 1 to Output 4, IDs = 1 to 4; see "I/O Socket" (p. 191).

<CTOPam> parameter IDs available for C-663:
1 = TriggerStep
2 = Axis
3 = TriggerMode
7 = Polarity
8 = StartThreshold
9 = StopThreshold
10 = TriggerPosition

<Value> available for the appropriate <CTOPam> ID:

for TriggerStep: step size in physical units (default value is 10.0).

for Axis: the identifier of the axis to be connected to the trigger output line. The axis identifier might have been changed with SAI (p. 138).

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: Should the motion direction be 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 written to the selected trigger output line (this status can also be

read with the ONT? command)
6 = InMotion; the selected trigger 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 increment value 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-663.

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

for StartThreshold/StopThreshold: position value in physical units;
if used for the PositionDistance trigger mode, both thresholds must be set to determine the position range and the direction of motion for trigger output;
StopThreshold is used as stop condition for Position+Offset trigger mode;
default values are -2147483647.0 and 2147483647.0

for TriggerPosition: position where the first trigger pulse is to be output in the Position+Offset trigger modes; in physical units (default value is 0.0)

For application examples and further details see "Configuring Trigger Output" (p. 40) 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:

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

Example 2: In this example, the trigger output line 1 shall be set from low to high when axis A starts to move. The following parameters must be set:
TrigOutID = 1
Axis = A (axis identifier was changed with SAI)
TriggerMode = 6
Polarity = Active High
So you have to send:
CTO 1 2 A
CTO 1 3 6
CTO 1 7 1

Example 3: The stage is connected to axis 1. The reference position of the stage is 12.5 mm. Starting from its reference position, the axis is to be moved alternating forwards and backwards, and 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. For further macro details see "Working with Controller Macros" (p. 59).
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 controller cycle of the C-663.

In this example, *Stepsize* is set to 0.02 mm so that the axis velocity should not exceed 200 mm/s.

```
mac beg TRIGREF
CTO 1 3 7
SVO 1 1
FRF
CTO 1 1 0.02
CTO 1 9 14.5
CTO 1 10 13.5
TRO 1 1
mac end

mac beg TRIGGER
CTO 1 1 0.02
CTO 1 9 14.5
CTO 1 10 13.5
DEL 1000
MOV 1 15.5
WAC POS? 1 > 15.3
MEX CTO? 1 10 < 14.4
CTO 1 1 -0.02
CTO 1 9 13.5
CTO 1 10 14.5
DEL 1000
MOV 1 12.5
WAC POS? 1 < 12.7
MEX CTO? 1 10 > 13.6
MAC START TRIGGER
mac end
```

CTO? (Get Configuration of Trigger Output)

Description: Replies with the values set for specified trigger output lines and parameters

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

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

<CTOPam>: parameter ID; see CTO.

If all arguments are omitted, the values for all parameters are given for all output lines.

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

For <Value> see CTO.

DEC (Set Deceleration)

Description: Set deceleration of given axes.

DEC can be changed while the axis is moving.

Format: DEC {<AxisID> <Deceleration>}

Arguments: <AxisID> is one axis of the controller

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

Response: none

Troubleshooting: Illegal axis identifiers

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

DEC changes the value of the Current deceleration parameter (ID 0xC in volatile memory (can be saved as default with WPA (p. 160), can also be changed with SPA (p. 142) and SEP (p. 139)).

The maximum value which can be set with the DEC command is given by the Maximum deceleration parameter, ID 0x4B (can be changed with SPA (p. 142) and SEP (p. 139)).

DEC? (Get Deceleration)

Description: Get the current value of the deceleration.

If all arguments are omitted, gets current value of all axes.

Format: DEC? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller

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

where

<float> is the current active deceleration value 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: See the MAC command (p. 124) and "Working with Controller Macros" (p. 59) for more information.

DIO (Set Digital Output Line)

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

Use TIO? (p. 151) to get the number of installed digital I/O lines.

Format: DIO {<DIOID> <OutputOn>}

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

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

Response: none

Notes: Using the DIO command, you can activate/deactivate the Output 1 to Output 4 lines which are located on the "I/O" socket (p. 191). With the C-663, you can either set a single line per DIO command, or all lines at once.

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

If <OutputOn>=1 the line is set to HIGH/ON, if <OutputOn>=0 it is set to LOW/OFF. If <DIOID> = 0, a bit pattern can be set in decimal or hexadecimal format which gives the states for all lines.

Do not use DIO on output lines for which the trigger output is activated with TRO (p. 153).

DIO? (Get Digital Input Lines)

Description: Lists the states of the specified digital input lines. Can be used to query externally generated signals.

Use TIO? (p. 151) to get the number of installed digital I/O lines.

Format: DIO? [{<DIOID>}]

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

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

where

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

Notes: Using the DIO? command, you can directly read the Input 1 to Input 4 lines which are located on the "I/O" socket (p. 191). With the C-663, you can either read a single line per DIO? command, or all lines at once.

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

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

DRC (Set Data Recorder Configuration)

Description:	Set data recorder configuration: determines the data source and the kind of data (RecordOption) used for the given data recorder table.
Format:	DRC <RecTableID> <Source> <RecOption>
Arguments:	<RecTableID>: is one data recorder table of the controller, see below. <Source>: is the data source, for example an axis, output signal channel or input signal channel of the controller. The required source depends on the selected record option. <RecOption>: is the kind of data to be recorded (record option). See below for a list of the available record options and the corresponding data sources.
Response:	none
Notes:	The C-663 has two data recorder tables with 1024 points per table. With HDR? (p. 111) 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. 153). For detailed information see "Data Recording" (p. 64).
Available record options with the appropriate data sources:	0=Nothing is recorded 1=Commanded position of axis 70=Commanded velocity of axis 71=Commanded acceleration of axis 73=Motor output of axis 74=External encoder of axis 80=Signal status register of axis 81=Analog input (channel = 1 - 9) The input channels can be the Input 1 to 4 lines of the I/O socket (p. 191) (use source ID 1 to 4) and the joystick axis (source ID 5) and joystick button inputs (source ID 6) on the Joystick socket (p. 193). Source ID 7 and above are reserved for additional analog input channels.

DRC? (Get Data Recorder Configuration)

Description: Returns settings made with DRC (p. 98).

Format: DRC? [{<RecTableID>}]

Arguments: <RecTableID>: is one data recorder table of the controller; if omitted settings for all tables are given.

Response: The current DRC settings:

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

where

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

<RecOption>: is the kind of data to be recorded

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

DRR? (Get Recorded Data Values)

Description: Reading of the last recorded Data Set.

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

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

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

Arguments: <StartPoint>: is the start point in the data recorder table, starts with index 1

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

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

Response: The recorded data in GCS array format, see the separate manual for GCS array, SM146E, and the example below.

Notes: If <RecTableID> is omitted, the data from all tables with non-zero record option (see DRC (p. 98)) is read.

With HDR? (p. 111) you will obtain a list of available record options and trigger options and additional information about data recording.

For detailed information see "Data Recording" (p. 64).

Example:

```
rtr?
10
drr? 1 20
# REM C-663
#
# VERSION = 1
# TYPE = 1
# SEPARATOR = 32
# DIM = 1
# SAMPLE_TIME = 0.000500
# NDATA = 20
#
# NAME0 = Commanded Position of Axis  AXIS:1
#
# END_HEADER
50.0000
50.0000
50.0000
50.0000
50.0000
50.0002
50.0002
50.0002
50.0003
50.0003
50.0003
50.0005
50.0005
50.0006
50.0008
50.0008
50.0009
50.0011
50.0013
50.0014
```

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. At present, only 0 is valid, which means that the specified trigger source is set for all data recorder tables. <TriggerSource> ID of the trigger source, see below for a list of available options <Value> depends on the trigger source, can be a dummy, see below.
Response:	none
Notes:	By default, data recording is triggered when a step response measurement is made with STE (p. 147). A trigger option set with DRT will become valid for all data recorder tables with non-zero record option (see DRC (p. 98)). With HDR? (p. 111) you will obtain a list of available record options and trigger options and additional information about data recording. For detailed information see "Data Recording" (p. 64).
Available trigger options:	0 = default setting; data recording is triggered with STE; <Value> must be a dummy 1 = any command changing target position (e.g. MVR (p. 131), MOV (p. 130)); <Value> must be a dummy 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

DRT? (Get Data Recorder Trigger Source)

Description:	Returns the trigger source for the data recorder tables.
Format:	DRT? [{<RecTableID>}]
Arguments:	<RecTableID> is one data recorder table of the controller
Response:	{<RecTableID>"="<TriggerSource> <Value> LF} where <TriggerSource> is the ID of the trigger source, see DRT (p. 101) for details. <Value> depends on the trigger source, if 0 it is a dummy, see DRT for details.
Notes:	Since all data recorder tables of the C-663 have the same trigger source, the DRT? response is given as a single line of the form 0=<TriggerSource> <Value>

ERR? (Get Error Number)

Description:	Get error code <int> of the last occurred error and reset the error to 0. Only the last error is buffered. Therefore you should call ERR? after each command. The error codes and their descriptions are fully listed in "Error Codes" (p. 162).
Format:	ERR?
Arguments:	none
Response:	The error code of the last occurred error (int).
Troubleshooting:	Communication breakdown

FED (Find Edge)

Description: Moves given axis to a given signal edge.

The motor must be switched on with SVO (p. 149) for the commanded axis prior to using this command.

In contrast to the referencing commands (FNL (p. 105), FPL (p. 107) and FRF (p. 108)), this command does not change the reference state of the axis and does not set a certain position value at the selected edge. It does move out of the limit condition, therefore the axis motion finishes at the same position as with the corresponding referencing commands.

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

<EdgeID> is the type of edge the axis has to move to. See below for available edge types.

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

Response: none

Troubleshooting: Illegal axis identifier; limit switches and/or reference switch are disabled (see below); the motor is switched off.

Notes:

The C-663 firmware detects the presence or absence of a built-in reference switch and built-in limit switches using controller parameters (ID 0x14 for reference switch; ID 0x32 for limit switches). According to the values of those parameters, the C-663 enables or disables FED motions to the appropriate signal edges. Adapt the parameter values to your hardware using SPA (p. 142) or SEP (p. 139). See "Controller Parameters" (p. 27) for more information.

You can use the digital input lines instead of the switches as source of the signals for FED. See parameters 0x5C (p. 31), 0x5D (p. 32) and 0x5E (p. 33) for details.

The digital input line selected as source of the reference signal is used for FED irrespective of the setting of parameter 0x14 (“Stage has a built-in reference switch”).

While the built-in limit switches of the stage are processed only if enabled with parameter 0x32 (“stage has limit switches”), the setting of parameter 0x32 has no influence on the usage of activated digital input lines.

FED can be used to measure the physical travel range of a new mechanical system and thus to identify the values for the corresponding controller parameters: the distance from negative to positive limit switch, the distance between the negative limit switch and the reference switch (DISTANCE_REF_TO_N_LIM, parameter ID 0x17), and the distance between reference switch and positive limit switch (DISTANCE_REF_TO_P_LIM, parameter ID 0x2F). See "Travel Range Adjustment" (p. 68) for more information.

The motion can be interrupted by #24 (p. 84), STP (p. 148) and HLT (p. 112).

Motion commands like FED are not allowed when the joystick is active for the axis. See "Joystick Control" (p. 56) for details.

Available edge types and parameters:	The following edge types with their parameter settings are available: 1 = negative limit switch, <Param> must be 0 2 = positive limit switch, <Param> must be 0 3 = reference switch, <Param> must be 0

FNL (Fast Reference Move To Negative Limit)

Description:	Performs a reference move. If successful, absolute motion will afterwards be possible. 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. 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 affected.
Response:	none
Troubleshooting:	Illegal axis identifier
Notes:	The motor must be switched on with SVO (p. 149) for the commanded axis prior to using this command. The reference mode must be set to "1" (factory default) with the RON command (p. 135) if referencing is to be done by performing a reference move. See "Referencing" (p. 37) for further details. With the C-663, the negative limit switch of the mechanical system is used to determine the negative physical limit of the travel range. The difference of VALUE_AT_REF_POS (parameter ID 0x16) and DISTANCE_REF_TO_N_LIM (parameter ID 0x17) is set as the current position when the axis is at the negative limit switch (value can be negative).

You can use a digital input instead of the negative limit switch as source of the negative limit switch signal for FNL. See parameter 0x5D (p. 32) for details.

This command can be interrupted by #24 (p. 84), STP (p. 148) and HLT (p. 112).

Use FRF? (p. 110) to check whether the reference move was successful.

Use FRF (p. 108) instead of FNL to perform a reference move for an axis which has no limit switches but a reference sensor.

For best repeatability, always reference in the same way.

If the soft limits (MAX_TRAVEL_RANGE_POS and MAX_TRAVEL_RANGE_NEG) 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 switch can be used for a reference move (FRF).

The soft limits may not be outside of the physical travel range:
MAX_TRAVEL_RANGE_POS ≤ DISTANCE_REF_TO_P_LIM + VALUE_AT_REF_POS
MAX_TRAVEL_RANGE_NEG ≥ VALUE_AT_REF_POS - DISTANCE_REF_TO_N_LIM
Otherwise, reference moves to the limit switches would have incorrect results because the values of the soft limits would be set at the end of the referencing procedure.

See "Travel Range Adjustment" (p. 68) for more information.

FPL (Fast Reference Move To Positive Limit)

Description: Performs a reference move. If successful, absolute motion will afterwards be possible.

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.

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 affected.

Response: none

Troubleshooting: Illegal axis identifier

Notes: The motor must be switched on with SVO (p. 149) for the commanded axis prior to using this command.

The reference mode must be set to "1" (factory default) with the RON command (p. 135) if referencing is to be done by performing a reference move. See "Referencing" (p. 37) for further details.

With the C-663, the positive limit switch of the mechanical system is used to determine the positive physical limit of the travel range. The sum of VALUE_AT_REF_POS (parameter ID 0x16) and DISTANCE_REF_TO_P_LIM (parameter ID 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. See parameter 0x5E (p. 33) for details.

This command can be interrupted by #24 (p. 84), STP (p. 148) and HLT (p. 112).

Use FRF? (p. 110) to check whether the reference move was successful.

Use FRF (p. 108) instead of FPL to perform a reference move for an axis which has no limit switches but a reference sensor.

For best repeatability, always reference in the same way.

If the soft limits (MAX_TRAVEL_RANGE_POS and MAX_TRAVEL_RANGE_NEG) 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 switch can be used for a reference move (FRF).

The soft limits may not be outside of the physical travel range:
MAX_TRAVEL_RANGE_POS ≤ DISTANCE_REF_TO_P_LIM + VALUE_AT_REF_POS
MAX_TRAVEL_RANGE_NEG ≥ VALUE_AT_REF_POS - DISTANCE_REF_TO_N_LIM
Otherwise, reference moves to the limit switches would have incorrect results because the values of the soft limits would be set at the end of the referencing procedure.

See "Travel Range Adjustment" (p. 68) for more information.

FRF (Fast Reference Move To Reference Switch)

Description: Performs a reference move. If successful, absolute motion will afterwards be possible.

Moves the given axis to the reference switch and sets the current position to a defined value. See below for details.

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 affected.

Response: none

Troubleshooting: Illegal axis identifier

Notes: The motor must be switched on with SVO (p. 149) for the commanded axis prior to using this command.

The reference mode must be set to "1" (factory default) with the RON command (p. 135) if referencing is to be done by performing a reference move. See "Referencing" (p. 37) for further details.

The value of the VALUE_AT_REF_POS parameter (ID 0x16) is set as the current position when the axis is at the reference switch.

You can use a digital input instead of the reference sensor as source of the reference signal for the FRF command. See parameter 0x5C (p. 31) for details.

This command can be interrupted by #24 (p. 84), STP (p. 148) and HLT (p. 112).

Use FRF? (p. 110) to check whether the reference move was successful.

Use FNL (p. 105) or FPL (p. 107) instead of FRF to perform a reference move for an axis which has no reference sensor but limit switches.

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

See "Travel Range Adjustment" (p. 68) for more information.

FRF? (Get Referencing Result)

Description: Indicates whether the given axis is referenced or not.

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. 105), FPL (p. 107) or FRF (p. 108) or when the position was set directly with POS (p. 133) (depending on the referencing mode set with RON (p. 135)).

Format: FRF? [{<AxisID>}]

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

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

where

<uint> indicates whether the axis was successfully referenced (=1) or not (=0).

Troubleshooting: Illegal axis identifier

GOH (Go To Home Position)

Description: Move given axes to home position.

GOH [{<AxisID>}]
is the same as
MOV {<AxisID> 0}

The motor must be switched on with SVO (p. 149) for the commanded axis prior to using this command.

This command can be interrupted by #24 (p. 84), STP (p. 148) and HLT (p. 112).

Format: GOH [{<AxisID>}]

Arguments: <AxisID>: is one axis of the controller, if omitted, all axes are affected

Response: none

Troubleshooting: Illegal axis identifier

HDR? (Get All Data Recorder Options)

Description: List 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
{<RecordOption>"="<DescriptionString>[of <Channel>]}

#TriggerOptions
[{<TriggerOption>"="<DescriptionString>}]

#Parameters to be set with SPA
[{<ParameterID>“=“<DescriptionString>}]

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

end of help

Example: For the C-663, the HDR? response is as follows:

```
#RecordOptions
0=Nothing is recorded
1=Commanded Position of Axis
70=Commanded Velocity of Axis
71=Commanded Acceleration of Axis
73=Motor Output of Axis
74=External Encoder of Axis
80=Signal Status Register of Axis
81=Analog input (Channel = 1 - 9)
#TriggerOptions
0=default setting
1=any command changing position (e.g. MOV)
2=next command
6=any command changing position (e.g. MOV), reset
trigger after execution
#Additional information
2 record tables
1024 datapoints per table
end of help
```

TriggerOptions = 0 (default) means that recording is triggered by the STE command (p. 147).

HLP? (Get List Of Available Commands)

Description:	List 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:	Halt the motion of given axes smoothly. For details see the notes below. Error code 10 is set. #24 (p. 84) and STP (p. 148) in contrast abort current motion as fast as possible for the controller without taking care of systems inertia or oscillations.
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. HLT stops all motion caused by move commands (e.g. MOV (p. 130), MVR (p. 131), GOH (p. 110), STE (p. 147)), referencing commands (FNL (p. 105), FPL (p. 107), FRF (p. 108)) and macros (MAC (p. 124)). 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. See "Controller Parameters" (p. 27) for further details.

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

SPA (p. 142) affects the parameter settings in volatile memory (RAM).

WPA (p. 160) copies parameter settings from RAM to non-volatile memory.

SEP (p. 139) writes parameter settings directly into non-volatile memory (without changing RAM settings).

RPA (p. 136) resets RAM to the values from non-volatile memory.

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<FunctionGroup
Description>TAB<ParameterDescription>[{TAB<PossibleV
alue>"="<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-663, 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

JAS? (Query Joystick Axis Status)

Description: Get the current status of the given axis of the given joystick device which is directly connected to the controller.

Format: JAS? [{<JoystickID> <JoystickAxis>}]

Arguments: <JoystickID> is one joystick device connected to the controller; see below for details.

<JoystickAxis> is one of the axes of the joystick device; see below for details.

Response: {<JoystickID> <JoystickAxis>"="<Amplitude>}

where

<Amplitude> is the factor which is currently applied to the current valid velocity setting of the controlled motion axis, corresponds to the current displacement of the joystick axis. See below for details.

Notes: One joystick device can be connected to the Joystick socket (p. 193) of the C-663, the identifier is 1. The C-663 supports one axis of the joystick device, the identifier of the joystick axis is 1. See also "Accessible Items and Their Identifiers" (p. 51).

The <Amplitude> factor is applied to the velocity set with VEL (p. 158), 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. 117) for an example).

JAX (Set Axis Controlled By Joystick)

Description: Set axis controlled by a joystick which is directly connected to the controller.

Each axis of the controller can only be controlled by one joystick axis.

Format: JAX <JoystickID> <JoystickAxis> <AxisID>

Arguments: <JoystickID> is one joystick device connected to the controller; see below for details

<JoystickAxis> is one of the axes of the joystick device; see below for details

<AxisID> is one axis of the controller

Response: none

Notes: One joystick device can be connected to the Joystick socket (p. 193) of the C-663, the identifier is 1. The C-663 supports one axis of the joystick device, the identifier of the joystick axis is 1. See also "Accessible Items and Their Identifiers" (p. 51).

JAX? (Get Axis Controlled By Joystick)

Description: Get axis controlled by a joystick which is directly connected to the controller.

Format: JAX? [{<JoystickID> <JoystickAxis>}]

Arguments: <JoystickID> is one joystick device connected to the controller; see below for details

<JoystickAxis> is one of the axes of the joystick device; see below for details

Response: {<JoystickID> <JoystickAxis>"="{<AxisID> }LF}

where

<AxisID> is one axis of the controller

Notes: One joystick device can be connected to the Joystick socket (p. 193) of the C-663, the identifier is 1. The C-663 supports one axis of the joystick device, the identifier of the joystick axis is 1. See also "Accessible Items and Their Identifiers" (p. 51).

JBS? (Query Joystick Button Status)

Description: Get the current status of the given button of the given joystick device which is directly connected to the controller.

Format: JBS? [{<JoystickID> <JoystickButton>}]

Arguments: <JoystickID> is one joystick device connected to the controller; see below for details.

<JoystickButton> is one of the buttons of the joystick device; see below for details.

Response: {<JoystickID> <JoystickButton> "="<State>}

where

<State> indicates if the joystick button is pressed; 0 = not pressed, 1 = pressed

Notes: One joystick device can be connected to the Joystick socket (p. 193) of the C-663, the identifier is 1. The C-663 supports one button of the joystick device, the identifier of the joystick button is 1. See also "Accessible Items and Their Identifiers" (p. 51).

JDT (Set Joystick Default Lookup Table)

Description: Set lookup table type for the given axis of the given joystick device which is directly connected to the controller.

The current valid lookup table content for the specified joystick axis is overwritten by the selection made with JDT.

Format: JDT {<JoystickID> <JoystickAxis> <uint>}

Arguments: <JoystickID> is one joystick device connected to the controller; see below for details.

<JoystickAxis> is one of the axes of the joystick device; see below for details.

<uint> defines the type of lookup table profile to use; see below for details.

Response: none

Notes:	One joystick device can be connected to the Joystick socket (p. 193) of the C-663, the identifier is 1. The C-663 supports one axis of the joystick device, the identifier of the joystick axis is 1. See also "Accessible Items and Their Identifiers" (p. 51).
Available lookup tables:	The C-663 provides the following types for the lookup table profile: 1 = linear (default) 2 = parabolic Use parameter 0x61 (p. 35) to invert the direction of motion.

JLT (Fill Joystick Lookup Table)

Description:	Fill the lookup table for the given axis of the given joystick device which is directly connected to the controller. The amplitudes of the joystick axes (i.e. their displacements) are mapped to the current valid velocity settings of the controller axes. For each joystick axis there is a lookup table that defines this mapping. With JLT this table can be written, or a default table profile provided by the controller can be loaded with the JDT command (p. 116).
Format:	JLT <JoystickID> <JoystickAxis> <Addr> <floatn>
Arguments:	<JoystickID> is one joystick device connected to the controller; see below for details. <JoystickAxis> is one of the axes of the joystick device; see below for details. <Addr> is the index of a point in the lookup table, starts with 1. <floatn> is the value of point n (n is 1 to 256). The first point corresponds to the maximum joystick axis displacement in negative direction, the 256th point to the maximum displacement in positive direction.
Response:	none

Notes: One joystick device can be connected to the Joystick socket (p. 193) of the C-663, the identifier is 1. The C-663 supports one axis of the joystick device, the identifier of the joystick axis is 1. See also "Accessible Items and Their Identifiers" (p. 51).

The <floatn> values are factors which will during joystick control be applied to the velocity set with VEL (p. 158), the range is -1.0000 to 1.0000.

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.

For further information see "Trajectory Generation" (p. 53).

JLT? (Get Joystick Lookup Table Values)

Description: Reading of the current valid lookup table values.

Format: JLT? <StartPoint> <NumberOfPoints> [{<JoystickID> <JoystickAxis>}]

Arguments: <StartPoint>: is the start point in the lookup table, starts with 1.

<NumberOfPoints>: is the number of points to be read per joystick axis; maximum number is 256.

<JoystickID> is one joystick device connected to the controller; see below for details.

<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, SM146E, and the example below.

Notes: With the C-663, <JoystickID> and <JoystickAxis> must be omitted in the JLT? command, while <StartPoint> and <NumberOfPoints> are always required.

The <floatn> values in the lookup table are factors which will during joystick control be applied to the velocity set with VEL (p. 158), 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: Enable or disable a joystick device which is directly connected to the controller.

For joystick control of a controller axis, this axis must be assigned to a joystick axis with JAX (p. 115), and the corresponding joystick device must be enabled with JON.

Format: JON {<JoystickID> <uint>}

Arguments: <JoystickID> is one joystick device connected to the controller; see below for details.

<uint> 1 enables the joystick device, 0 disables joystick device.

Response: none

Notes: One joystick device can be connected to the Joystick socket (p. 193) of the C-663, the identifier is 1. See also "Accessible Items and Their Identifiers" (p. 51).

Before a joystick device can be activated with JON, its axes must have been assigned to the controller axes using JAX (p. 115).

While a joystick connected to the C-663 is enabled with the JON command, this joystick controls the axis velocity ("commanded velocity" output by the trajectory generator). During joystick control, the target position is set to the travel range limits given by the Max_Travel_Range_pos and Max_Travel_Range_neg parameters (0x15 and 0x30, see "Travel Range Adjustment" (p. 68) for more information). For joystick operation, the motor of the stage must be switched on with SVO. When disabling a joystick, the target position is set to the current position for joystick-controlled axes.

Motion commands like MOV (p. 130) are not allowed when a joystick is active on the axis. See "Joystick Control" (p. 56) for details.

JON? (Get Joystick Activation Status)

Description:	Get activation state of the given joystick device which is directly connected to the controller.
Format:	JON? [{<JoystickID>}]
Arguments:	<JoystickID> is one joystick device connected to the controller; see below for details.
Response:	{<JoystickID>"="<uint>} where <uint> is the joystick activation state: 1 = joystick device enabled, 0 = joystick device disabled.
Notes:	One joystick device can be connected to the Joystick socket (p. 193) of the C-663, the identifier is 1. See also "Accessible Items and Their Identifiers" (p. 51).

JRC (Jump Relatively Depending on Condition)

Description: **J**ump **R**elatively depending on a given **C**ondition of the following type: one given value is compared with a queried value according to a given rule.
Can only be used in macros.

Format: JRC <Jump> <CMD?> <OP> <Value>

Arguments:

<Jump> is the size of the relative jump. -1 means the macro execution pointer goes to the previous line, 0 means the command is executed again, which is the same behaviour as WAC. 1 goes 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. Which commands are supported by a firmware is controller specific.

<OP> is the operator to be used. The following are possible: = <= < > >= !=.
Which operators are supported by a firmware is controller-specific.

<Value> is the value to be compared with the response of <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 result of the POS? 1 query is being copied to the variable TARGET. 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 then defined empty with the VAR command which deletes the variable.

Write macro “stop”:

```
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:	Indicates whether axes have built-in limit switches.
Format:	LIM? [{<AxisID>}]
Arguments:	<AxisID>: is one axis of the controller
Response:	{<AxisID>"="<uint> LF} where <uint> indicates whether the axis has built-in limit switches (=1) or not (=0).
Troubleshooting:	Illegal axis identifier
Notes:	The C-663 firmware detects the presence or absence of built-in limit switches using a controller parameter (ID 0x32). According to the value of this parameter, the C-663 enables or disables the stopping of the motion at the built-in limit switches and reference moves to those limit switches using FNL (p. 105) or FPL (p. 107). You can use the digital input lines as source of the negative or positive limit signal. See parameters 0x5D (p. 32) and 0x5E (p. 33) for details. The setting of parameter 0x32 has no influence on the usage of activated digital input lines. Adapt the parameter value to your hardware using SPA (p. 142) or SEP (p. 139). See "Controller Parameters" (p. 27) for more information.

MAC (Call Macro Function)

Description: Call a macro function. Permits recording, deleting and running macros on the controller.

Format: MAC <keyword> {<parameter>}

in particular:

MAC BEG <macroname>
MAC DEF <macroname>
MAC DEF?
MAC DEL <macroname>
MAC END
MAC ERR?
MAC NSTART <macroname> <uint> [<String1> [<String2>]]
MAC START <macroname> [<String1> [<String2>]]

Arguments: <keyword> determines which macro function is called. The following keywords and parameters are used:

MAC BEG <macroname>
Start recording a macro to be named *macroname* on the controller; may not be used in a macro; the commands that follow become the macro, so if successful, the error code cannot be queried. End the recording with MAC END.

MAC ERR?
Reports the first error which occurred during macro execution.
Response: <macroname> <uint1>"="<unit2> <"<"CMD">">
where <macroname> is the name of the macro, <unit1> is the line in the macro, <unit2> is the error code and <"<"CMD">"> is the erroneous command which was sent to the parser.

MAC END
Stop macro recording (cannot become part of a macro)

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

MAC DEF?
Ask 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>]]
Execute 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 the local variables 1 and 2 used in the specified 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. 76) for more information.

MAC START <macroname> [<String1> [<String2>]]
Starts one execution of 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).
Macro contains a disallowed MAC command.

Notes: During macro recording no macro execution is allowed.

When a macro is recorded for a controller whose address is different from 1, the target ID must be part of each command line, but will not become part of the macro content. See "Defining Macros" (p. 59) and "Target and Sender Address" (p. 75) for more information.

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

Macros can contain local and global variables. See “Variables” (p. 76) for details.

A running macro sends no responses to any interface. This means questioning commands are allowed in macros but not answered and therefore useless.

Depending on the value of parameter 0x72 (Ignore Macro Error), there are the following possibilities when an error is caused by the running macro:

- 0 = macro execution is aborted
- 1 = the error is ignored and macro execution will be continued

Irrespective of the parameter setting, MAC ERR? always reports the first error which occurred during macro execution.

The following commands provided by the C-663 can only be used in macros:
DEL (p. 96), JRC (p. 122), MEX (p. 128) and WAC (p. 159).

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 move commands received from the command line may overwrite each other, and only the last move command will be executed, irrespective of its source.

Macro execution can be stopped with #24 (p. 84), STP (p. 148) and HLT (p. 112).

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

A running macro may not be deleted.

Macro execution is not allowed when a joystick is active on the axis, but macro recording is possible. See "Joystick Control" (p. 56) for details.

You can query with #8 (p. 83) if a macro is currently running on the controller.

Warning: The number of write cycles of non-volatile memory is limited.

MAC? (List Macros)

Description: List 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>
if <macroname> was given, <string> is the content of this macro;
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:	Stop macro execution due to a given condition of the following type: a specified value is compared with a queried value according to a specified rule. Can only be used in macros. When the macro interpreter accesses this command the condition is checked. If it is true the current macro is stopped, otherwise macro execution is continued with the next line. Should the condition be fulfilled later, the interpreter will ignore it. See also WAC (p. 159).
Format:	MEX <CMD?> <OP> <value>
Arguments:	<CMD?> is one questioning command in its usual syntax. The answer must consist of a single value. For an example see below. <OP> is the operator to be used. The following operators are possible: = <= < > >= != <value> is the value to be compared with the response of <CMD?>
Response:	none

Example: Send: MAC START AMC001

Note: Macro "AMC001" has the following contents:

MAC START AMC002
MAC START AMC003
MEX DIO? 4 = 1
MAC START AMC001

Macro " AMC002" has the following contents:

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

Macro AMC003" has the following content:

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

Macro AMC001 forms an infinite loop by permanently calling AMC002, AMC003 and itself.

AMC002 checks the state of the digital input channel 1 (located on the I/O socket (p. 191)). 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).

AMC003 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.

The motor must be switched on with SVO (p. 149) for the commanded axis prior to using this command.

Format: MOV {<AxisID> <Position>}

Arguments <AxisID> is one axis of the controller

<Position> is the new absolute target position in physical units.

Response: none

Notes: The target position must be inside the travel range limits. Use TMN? (p. 152) and TMX? (p. 152) to ask for the current valid travel range limits.

The motion can be interrupted by #24 (p. 84), STP (p. 148) and HLT (p. 112).

During a move, a new move command resets the target to a new value and the old one may never be reached. This is also valid with macros: move commands can be sent from the command line when a macro is running. The macro content and move commands received from the command line may overwrite each other, and only the last move command will be executed, irrespective of its source.

Motion commands like MOV are not allowed when a joystick is active on the axis. See "Joystick Control" (p. 56) for details.

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 reply to the ERR? command (p. 102) indicates that the target position given in the move command is out of limits.

MOV? (Get Target Position)

Description:	Returns last valid commanded target position.
Format:	MOV? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller
Response:	{<AxisID>"="<float> LF} where <float> is the last commanded target position in physical units
Troubleshooting:	Illegal axis identifier
Notes:	The target position can be changed by commands that cause motion (e.g. MOV (p. 130), MVR (p. 131), GOH (p. 110), STE (p. 147)) or by the joystick (when disabling a joystick, the target position is set to the current position for joystick-controlled axes). MOV? gets the commanded positions. Use POS? (p. 134) to get the current positions.

MVR (Set Target Relative To Current Position)

Description:	Move given axes relative to the last commanded target position. The new target position is calculated by adding the given value <Distance> to the last commanded target value. The motor must be switched on with SVO (p. 149) for the commanded axis prior to using this command.
Format:	MVR {<AxisID> <Distance>}
Arguments:	<AxisID> is one axis of the controller. <Distance> gives the distance 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 travel range limits. Use TMN? (p. 152) and TMX? (p. 152) to ask for the current valid travel range limits, and MOV? (p. 131) for the current target.
	The motion can be interrupted by #24 (p. 84), STP (p. 148) and HLT (p. 112).
	During a move, a new move command resets the target to a new value and the old one may never be reached. This is also valid with macros: move commands can be sent from the command line when a macro is running. The macro content and move commands received from the command line may overwrite each other, and only the last move command will be executed, irrespective of its source.
	Motion commands like MVR are not allowed when a joystick is active on the axis. See "Joystick Control" (p. 56) for details.

Example:

Send:	MOV 1 0.5
Note:	This is an absolute move.
Send:	POS? 1
Receive:	1=0.500000
Send:	MOV? 1
Receive:	1=0.500000
Send:	MVR 1 2
Note:	This is a relative move.
Send:	POS? 1
Receive:	1=2.500000
Send:	MVR 1 2000
Note:	New target position of axis 1 would exceed motion range. Command is ignored, i.e. the target position remains unchanged, and the axis does not move.
Send:	MOV? 1
Receive:	1=2.500000
Send:	POS? 1
Receive:	1=2.500000

ONT? (Get On Target State)

Description: Get on-target status of given axis.

If all arguments are omitted, gets status of all axes.

Format: ONT? [{<AxisID>}]

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

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

where

<uint> = "1" when the specified axis is on-target, "0" otherwise.

Troubleshooting: Illegal axis identifier

Notes: The on-target status becomes true when the trajectory has finished.

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.

<Position> is the new current position in physical units.

Response: none

Troubleshooting: Illegal axis identifier

Notes: Setting the current position with POS is only possible when the reference mode is set to "0", see RON (p. 135).

An axis is considered as "referenced" when the position was set with POS (for more information refer to "Referencing" (p. 37)).

The minimum and maximum commandable positions (TMN? (p. 152), TMX? (p. 152)) are not adapted when a position is set with POS. This may result in target positions which are allowed by the software and cannot be reached by the hardware. Also target positions are possible which can be reached by the hardware but are denied by the software. Furthermore, the home position can be outside of the physical travel range after using POS.

POS? (Get Real Position)

Description: Returns the current axis position.

If all arguments are omitted, gets current position of all axes.

Format: POS? [{<AxisID>}]

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

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

where

<float> is the current axis position in physical units

Troubleshooting: Illegal axis identifier

RBT (Reboot System)

Description: Reboot system. Controller behaves just like after power-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:	List macros which are currently running.
Format:	RMC?
Arguments:	none
Response:	{<macroname> LF} where <macroname> is the name of one macro which is saved on the controller and currently running. The response is an empty line when no macro is running.

RON (Set Reference Mode)

Description:	Set reference mode of given axes.
Format:	RON {<AxisID> <ReferenceOn>}
Arguments:	<AxisID> is one axis of the controller. <ReferenceOn> can be 0 or 1: 0= referencing moves with FRF (p. 108), FNL (p. 105) and FPL (p. 107) are not possible, absolute position must be set with POS (p. 133) to reference the axis. 1= FRF or FNL or FPL is required to reference the axis, usage of POS is not allowed. 1 is default.
Response:	none
Troubleshooting:	Illegal axis identifier
Notes:	For more information refer to "Referencing" (p. 37) and "Travel Range Adjustment" (p. 68).

RON? (Get Reference Mode)

Description: Get reference mode of given axes.

Format: RON? [{ <AxisID>}]

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

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

where

<ReferenceOn> is the current reference mode of the controller, see RON (p. 135).

Troubleshooting: Illegal axis identifier

RPA (Reset Volatile Memory Parameters)

Description: Resets the given parameter of the given item. The value from non-volatile memory is written into volatile memory.

Related commands:

With HPA? (p. 113) you can obtain a list of the available parameters. SPA (p. 142) affects the parameter settings in volatile memory, WPA (p. 160) writes parameter settings from volatile to non-volatile memory, and SEP (p. 139) writes parameter settings directly into non-volatile memory (without changing the settings in volatile memory).

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.

<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-663, 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. 138). See "Accessible Items and Their Identifiers" (p. 51) for more information. Valid parameter IDs are given in "Controller Parameters" (p. 27).

RTR (Set Record Table Rate)

Description:	Sets the record table rate, i.e. the number of controller-loop 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 table rate to be used for recording operations (unit: number of controller-loop cycles), must be an integer value larger than zero
Response:	None
Notes:	The duration of the recording can be calculated as follows: Rec. Duration = Controller Cycle Time * RTR value * Number of Points where Controller Cycle Time is 50 µs for the C-663 Number of Points is 1024 for the C-663 (length of data recorder table) For more information see "Data Recording" (p. 64). 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 controller-loop cycles used in data recording operations.

Format: RTR?

Arguments: None

Response: <RecordTableRate> is the table rate used for recording operations (unit: number of controller-loop 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

<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. 155) to ask for valid characters.

The new axis identifier is saved automatically and thus still available after reboot or next power-on.

SAI? (Get List Of Current Axis Identifiers)

Description: Gets the axis identifiers.

See also "Accessible Items and Their Identifiers" (p. 51).

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}

<AxisID> is one axis of the controller.

SEP (Set Non-Volatile Memory Parameters)

Description: Set a parameter of a given item to a different value in non-volatile memory, where it becomes the new power-on default.

After parameters were set with SEP, you can use RPA (p. 136) to activate them (write them to volatile memory) without controller reboot.

Caution: This command is for setting hardware-specific parameters. Wrong values may lead to improper operation or damage of your hardware!

Related commands:

HPA? (p. 113) returns a list of the available parameters.

SPA (p. 142) writes parameter settings into volatile memory (without changing the settings in non-volatile memory).

WPA (p. 160) writes parameter settings from volatile to non-volatile memory.

See SPA for an example.

Format: SEP <Pswd> {<ItemID> <PamID> <PamValue>}

Arguments	<Pswd> is the password for writing to non-volatile memory, default is "100" <ItemID> is the item for which a parameter is to be changed in non-volatile memory. See below for details. <PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details. <PamValue> is the value to which the given parameter of the given item is set.
Response:	none
Troubleshooting:	Illegal item identifier, wrong parameter ID, invalid password
Notes:	Warning: The number of write cycles of non-volatile memory is limited. Write default values only when necessary. With the C-663, 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. 138). See "Accessible Items and Their Identifiers" (p. 51) for more information. Valid parameter IDs are given in "Controller Parameters" (p. 27).

SEP? (Get Non-Volatile Memory Parameters)

Description:	Get the value of a parameter of a given item from non-volatile memory. With HPA? (p. 113) 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 non-volatile memory is to be queried. See below for details. <PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details.

Response:	{<ItemID> <PamID>"="<PamValue> LF} where <PamValue> is the value of the given parameter for the given item.
Troubleshooting:	Illegal item identifier, wrong parameter ID
Notes:	With the C-663, 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. 138). See "Accessible Items and Their Identifiers" (p. 51) for more information. Valid parameter IDs are given in "Controller Parameters" (p. 27).

SPA (Set Volatile Memory Parameters)

Description: Set a parameter of a given item to a value in volatile memory (RAM). Parameter changes will be lost when the controller is powered down or rebooted or when the parameters are restored with RPA (p. 136).

Caution: This command is for setting hardware-specific parameters. Wrong values may lead to improper operation or damage of your hardware!

Related commands:

HPA? (p. 113) returns a list of the available parameters.

SEP (p. 139) writes parameter settings directly into non-volatile memory (without changing the settings in volatile memory).

WPA (p. 160) writes parameter settings from volatile to non-volatile memory.

RPA resets volatile memory to the value in non-volatile memory.

Format: SPA {<ItemID> <PamID> <PamValue>}

Arguments: <ItemID> is the item for which a parameter is to be changed in volatile memory. See below for details.

<PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details.

<PamValue> is the value to which the given parameter of the given item is set.

Response: none

Troubleshooting: Illegal item identifier, wrong parameter ID, value out of range

Notes: With the C-663, you can write only 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. 138). See "Accessible Items and Their Identifiers" (p. 51) for more information. Valid parameter IDs are given in "Controller Parameters" (p. 27).
Example 1:	Send: SPA 1 0x41 700 Note: Set the operating current for axis 1 to 700, parameter ID written in hexadecimal format. Send: SPA 1 64 140 Note: Set the holding current for axis 1 to 140, parameter ID written in decimal format.
Example 2:	The parameters must be adapted to a new load applied to the connected mechanical system. Send: SPA 1 0x41 900 Note: The operating current is set to 900 for axis 1. The setting is made in the volatile memory only. Now set the holding current and holding current delay in the volatile memory using SPA and then test the functioning of the system. If the system performance proves satisfactory and you want to use this system configuration after the next power-on, save the parameter settings from the volatile to the non-volatile memory. Send: WPA 100 Note: This command saves the values of all parameters whose password is 100 (see the list in "Controller Parameters" (p. 27)), since WPA is used without specifying any parameter.

Example 3:	Send:	SEP 100 LEFT 0x41 800
	Note:	The maximum operating current is to be set to 800 for axis LEFT (axis was renamed with SAI). The setting is made in the 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 the volatile memory (otherwise they would become active only after the next power-on or reboot of the controller).
	Send:	RPA
	Note:	The new configuration is now active.
	Send:	SPA? LEFT 0x41
	Receive:	LEFT 0X41=804 The difference between the sent value (800) and the received value (804) is due to the resolution of the D/A converter.
	Note:	Check the parameter settings in the volatile memory.

SPA? (Get Volatile Memory Parameters)

Description:	Get the value of a parameter of a given item from volatile memory (RAM). With HPA? (p. 113) you can obtain a list of the available parameters and their IDs.
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 ID, can be written in hexadecimal or decimal format. See below for details.
Response:	{<ItemID> <PamID>"="<PamValue> LF} where <PamValue> is the value of the given parameter for the given item.
Troubleshooting:	Illegal item identifier, wrong parameter ID
Notes:	With the C-663, 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. 138). See "Accessible Items and Their Identifiers" (p. 51) for more information. Valid parameter IDs are given in "Controller Parameters" (p. 27).

SRG? (Query Status Register Value)

Description: Returns register values for queried axes and register numbers.

Format: SRG? {<AxisID> <RegisterID>}

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

<RegisterID>: is the ID of the specified register, see below for available registers.

Response: {<AxisID><RegisterID>"="<Value> LF}

where

<Value> is the value of the register, see below for details.

Note: This command is identical in function to #4 (p. 82) 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
Description	On Target	Is referencing	Is Moving	Motor On	-	-	-	Error flag

Bit	7	6	5	4	3	2	1	0
Description	Digital Input 4	Digital Input 3	Digital Input 2	Digital Input 1	-	Positive Limit	Reference	Negative Limit

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 motor is switched 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 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. 98). The recorded data can be read with the DRR? (p. 99) command.
Format:	STE <AxisID> <Amplitude>
Arguments	<AxisID> is one axis of the controller. <Amplitude> is the height of the step. See below for details.
Response:	none
Troubleshooting:	The motor must be switched on with SVO (p. 149) for the commanded axis prior to using this command. The target position must be inside the travel range limits. Use TMN? (p. 152) and TMX? (p. 152) to ask for the current valid travel range limits, and MOV? (p. 131) for the current target. Motion commands like STE are not allowed when the joystick is active for the axis. See "Joystick Control" (p. 56) for details.
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 motion abruptly. For details see the notes below. Sets error code to 10. This command is identical in function to #24 (p. 84) which should be preferred when the controller is performing time-consuming tasks.
Format:	STP
Arguments:	none
Response:	none
Troubleshooting:	Communication breakdown
Notes:	STP stops all motion caused by move commands (e.g. MOV (p. 130), MVR (p. 131), GOH (p. 110), STE (p. 147)), referencing commands (FNL (p. 105), FPL (p. 107), FRF (p. 108)) and macros (MAC (p. 124)). After the axes are stopped, their target positions are set to their current positions. HLT (p. 112) in contrast to STP stops motion with given system deceleration with regard to system inertia.

SVO (Set Motor State)

Description:	Sets the motor state for given axes (motor on / motor off).
Format:	SVO {<AxisID> <MotorState>}
Arguments:	<AxisID> is one axis of the controller <MotorState> can have the following values: 0 = motor off 1 = motor on
Response:	None
Troubleshooting:	Illegal axis identifier
Notes:	The motor must be switched on with SVO before motions can be commanded. Use the motion commands MOV (p. 130), MVR (p. 131) and GOH (p. 110), or use the joystick control (p. 56). For reference moves with FRF (p. 108), FNL (p. 105) or FPL (p. 107), the motor must also be switched on. If the motor is switched off with SVO while the axis is moving, the axis stops and the target position is set to the current position. Using a start-up macro, you can configure the controller so that the motor is automatically switched on upon power-on or reboot. See "Start-Up Macro" (p. 62) for details.

SVO? (Get Motor State)

Description: Gets the motor state of given axes.

If all arguments are omitted, gets status of all axes.

Format: SVO? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller

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

where

<MotorState> is the current motor state of the axis:
0 = motor off
1 = motor on

Troubleshooting: Illegal axis identifier

TAC? (Tell Analog Channels)

Description: Get the number of installed analog lines.

Format: TAC?

Parameter: None

Response: <uint> gives the total number of analog lines.

Notes: Gets the number of analog input lines located on the I/O socket (p. 191) of the C-663 (Input 1 to Input 4). Note that these lines can also be used for digital input. See "Accessible Items and Their Identifiers" (p. 51) for more information.

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:	<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. 191) of the C-663. The identifiers of the lines are 1 to 4. See "Accessible Items and Their Identifiers" (p. 51) for more information. You can record the values of the analog input lines using the DRC record option 81 (p. 98).

TIO? (Tell Digital I/O Lines)

Description:	Tell number of installed digital I/O lines
Format:	TIO?
Arguments:	none
Response:	I=<uint1> O=<uint2> where <uint1> is the number of digital input lines. <uint2> is the number of digital output lines.
Notes:	The digital output lines reported by TIO? are Output 1 to Output 4. They can be set with DIO (p. 96). The digital input lines reported by TIO? are Input 1 to Input 4. The can be read with DIO? (p. 97), #4 (p. 82) and SRG? (p. 146). All the lines are located on the I/O socket (p. 191) of the C-663.

TMN? (Get Minimum Commandable Position)

Description: Get the minimum commandable position in physical units.

Format: TMN? [{ <AxisID>}]

Arguments: <AxisID> is one axis of the controller

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

where

<float> is the minimum commandable position in physical units.

Note: The minimum commandable position is defined by the MAX_TRAVEL_RANGE_NEG parameter, ID 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 MAX_TRAVEL_RANGE_POS parameter, ID 0x15.

TNR? (Get Number of Record Tables)

Description:	Get the number of data recorder tables currently available on the controller.
Format:	TNR?
Arguments:	none
Response	<uint> is the number of data recorder tables which are currently available
Notes:	The C-663 has 2 data recorder tables with 1024 data points per table. For more information see "Data Recording" (p. 64).

TRO (Set Trigger Output State)

Description:	Enables or disables the trigger output conditions which were set with CTO (p. 90) for the given trigger output line.
Format:	TRO {<TrigOutID> <TrigMode>}
Arguments:	<TrigOutID> is one digital output line of the controller, see below for details. <TrigMode> can have the following values: 0 = trigger output disabled 1 = trigger output enabled
Response:	none
Troubleshooting:	Illegal output line identifier
Notes:	<TrigOutID> corresponds to the output lines Output 1 to Output 4, IDs = 1 to 4; see "I/O Socket" (p. 191). Do not use DIO (p. 96) on output lines for which the trigger output is activated with TRO.

TRO? (Get Trigger Output State)

Description:	Gets enable status for given trigger output line (the trigger output configuration is made with CTO (p. 90)). If all arguments are omitted, gets status of all trigger output lines.
Format:	TRO? [{<TrigOutID>}]
Arguments:	<TrigOutID> is one digital output line of the controller, see TRO (p. 153) for details.
Response:	{<TrigOutID>"="<TrigMode> LF} where <TrigMode> is the current enable state of the trigger output line: 0 = trigger output disabled 1 = trigger output enabled
Troubleshooting:	Illegal output line identifier

TRS? (Indicate Reference Switch)

Description:	Indicates whether axes have a built-in reference sensor with direction sensing.
Format:	TRS? [{<AxisID>}]
Arguments:	<AxisID>: is one axis of the controller
Response:	{<AxisID>"="<uint> LF} where <uint> indicates whether the axis has a built-in direction-sensing reference sensor (=1) or not (=0).
Troubleshooting:	Illegal axis identifier
Notes:	The C-663 firmware detects the presence or absence of a built-in reference switch using a controller parameter (ID 0x14). According to the value of this parameter, the C-663 enables or disables reference moves to the built-in reference sensor of the stage (FRF command (p. 108)). Adapt the parameter value to your hardware using SPA (p. 142) or SEP (p. 139). See "Controller Parameters" (p. 27) for more information. You can use a digital input instead of the reference sensor as source of the reference signal for the FRF command. See parameter 0x5C (p. 31) for details. The digital input line selected as source of the reference signal is used for referencing irrespective of the setting of parameter 0x14.

TVI? (Tell Valid Character Set For Axis Identifiers)

Description:	Gets a string with characters which can be used for axis identifiers. Use SAI (p. 138) to change the axis identifiers and SAI? (p. 138) to ask for the current valid axis identifiers.
Format:	TVI?
Arguments:	none
Response:	<string> is a list of characters With the C-663, the string consists of 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ-_

VAR (Set Variable Value)

Description: Set a variable to a certain value.
Local variables can be set using VAR in macros only. See "Variables" (p. 76) for details regarding local and global variables.
The variable is present in RAM only.

Format: VAR <Variable> <String>

Arguments:

<Variable> is the name of the variable whose value is to be set.

<String> is the value to which the variable is to be set. If omitted, the variable is deleted.
The value can be given directly or via the value of a variable.

See "Variables" (p. 76) 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. 86) for another example.

VAR? (Get Variable Values)

Description:	Get variable values. Local variables can be queried using VAR? only if a macro with local variables is running. See “Variables” (p. 76) for details regarding local and global variables. If VAR? is combined with CPY, JRC, MEX or WAC, the response to VAR? has to be a single value and not more.
Format:	VAR? [{<Variable>}]
Arguments:	<Variable> is the name of the variable to be queried. See “Variables” (p. 76) for name conventions. If <Variable> is omitted, all global variables present in RAM are listed.
Response:	{<Variable>”=”<String>LF} where <String> gives the value to which the variable is set
Example:	See ADD (p. 86) for an example.

VEL (Set Velocity)

Description:	Set velocity of given axes. VEL can be changed while the axis is moving.
Format:	VEL {<AxisID> <Velocity>}
Arguments:	<AxisID> is one axis of the controller <Velocity> is the velocity value in physical units/s.
Response:	none
Troubleshooting:	Illegal axis identifiers
Notes:	The lowest possible value for <Velocity> is 0. VEL changes the value of the Current velocity parameter (ID 0x49) in volatile memory (can be saved as default with WPA (p. 160), can also be changed with SPA (p. 142) and SEP (p. 139)). The maximum value which can be set with the VEL command is given by the Maximum velocity parameter, ID 0xA (can be changed with SPA (p. 142) and SEP (p. 139)).

VEL? (Get Velocity)

Description:	Get the current velocity value. If all arguments are omitted, gets current value of all axes.
Format:	VEL? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller
Response:	{<AxisID>"="<float> LF} where <float> is the current active velocity value in physical units / s.
Notes:	VEL? queries the current value of the velocity.

VER? (Get Versions of Firmware and Drivers)

Description: Gets the versions of the C-663 firmware and the drivers and libraries used.

Format: VER?

Arguments: none

Response: {<string1>":" <string2> [<string3>]LF}

where

<string1> is the name of the component
<string2> is the version information of the component <string1>
<string3> is an optional note

WAC (Wait For Condition)

Description: Wait until a given condition of the following type occurs: a specified value is compared with a queried value according to a specified rule.

Can only be used in macros.

See also MEX (p. 128).

Format: WAC <CMD?> <OP> <value>

Arguments <CMD?> is one questioning command in its usual syntax. The answer must consist of a single value. For an example see below.

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

<value> is the value to be compared with the response of <CMD?>

Response: none

Example: Send: MAC BEG AMC028
MVR 1 1
WAC ONT? 1 = 1
MVR 1 -1
WAC ONT? 1 = 1
MAC START AMC028
MAC END
MAC START AMC028

Note: Macro AMC028 is recorded and then started. WAC ONT? 1 = 1 waits until the answer to ONT? 1 is 1=1. To form an infinite loop, the macro calls itself.

WPA (Save Parameters To Non-Volatile Memory)

Description: Write the currently valid value of a parameter of a given item from volatile memory (RAM) to non-volatile memory. The values saved this way become the default values.

Caution: If current parameter values are incorrect, the system may malfunction. Be sure that you have the correct parameter settings before using the WPA command.

RAM settings not saved with WPA will be lost when the controller is powered down or rebooted or when RPA (p. 136) is used to restore the parameters.

With HPA? (p. 113) you can obtain a list of all available parameters.

Use SPA? (p. 142) to check the current parameter settings in volatile memory.

See SPA (p. 142) for an example.

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

Arguments	<Pswd> is the password for writing to non-volatile memory. See below for details. <ItemID> is the item for which parameters are to be saved from volatile to non-volatile memory. See below for details. <PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details.
Response:	none
Troubleshooting:	Illegal item identifier, wrong parameter ID, invalid password
Notes:	Parameters can be changed in volatile memory with SPA (p. 142), ACC (p. 85), DEC (p. 95) and VEL (p. 158). 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. Warning: The number of write cycles of non-volatile memory is limited. Write default values only when 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. 138). See "Accessible Items and Their Identifiers" (p. 51) for more information. Valid parameter IDs are given in "Controller Parameters" (p. 27).

9.4 Error Codes

The error codes listed here are those of the PI General Command Set. As such, some may be not relevant to your controller and will simply never occur.

Controller Errors

0	PI_CNTR_NO_ERROR	No error
1	PI_CNTR_PARAM_SYNTAX	Parameter syntax error
2	PI_CNTR_UNKNOWN_COMMAND	Unknown command
3	PI_CNTR_COMMAND_TOO_LONG	Command length out of limits or command buffer overrun
4	PI_CNTR_SCAN_ERROR	Error while scanning
5	PI_CNTR_MOVE_WITHOUT_REF_OR_NO_SERVO	Unallowable move attempted on unreferenced axis, or move attempted with servo off
6	PI_CNTR_INVALID_SGA_PARAM	Parameter for SGA not valid
7	PI_CNTR_POS_OUT_OF_LIMITS	Position out of limits
8	PI_CNTR_VEL_OUT_OF_LIMITS	Velocity out of limits
9	PI_CNTR_SET_PIVOT_NOT_POSSIBLE	Attempt to set pivot point while U,V and W not all 0
10	PI_CNTR_STOP	Controller was stopped by command
11	PI_CNTR_SST_OR_SCAN_RANGE	Parameter for SST or for one of the embedded scan algorithms out of range
12	PI_CNTR_INVALID_SCAN_AXES	Invalid axis combination for fast scan
13	PI_CNTR_INVALID_NAV_PARAM	Parameter for NAV out of range
14	PI_CNTR_INVALID_ANALOG_INPUT	Invalid analog channel
15	PI_CNTR_INVALID_AXIS_IDENTIFIER	Invalid axis identifier
16	PI_CNTR_INVALID_STAGE_NAME	Unknown stage name
17	PI_CNTR_PARAM_OUT_OF_RANGE	Parameter out of range
18	PI_CNTR_INVALID_MACRO_NAME	Invalid macro name

19	PI_CNTR_MACRO_RECORD	Error while recording macro
20	PI_CNTR_MACRO_NOT_FOUND	Macro not found
21	PI_CNTR_AXIS_HAS_NO_BRAKE	Axis has no brake
22	PI_CNTR_DOUBLE_AXIS	Axis identifier specified more than once
23	PI_CNTR_ILLEGAL_AXIS	Illegal axis
24	PI_CNTR_PARAM_NR	Incorrect number of parameters
25	PI_CNTR_INVALID_REAL_NR	Invalid floating point number
26	PI_CNTR_MISSING_PARAM	Parameter missing
27	PI_CNTR_SOFT_LIMIT_OUT_OF_RANGE	Soft limit out of range
28	PI_CNTR_NO_MANUAL_PAD	No manual pad found
29	PI_CNTR_NO_JUMP	No more step-response values
30	PI_CNTR_INVALID_JUMP	No step-response values recorded
31	PI_CNTR_AXIS_HAS_NO_REFERENCE	Axis has no reference sensor
32	PI_CNTR_STAGE_HAS_NO_LIM_SWITCH	Axis has no limit switch
33	PI_CNTR_NO_RELAY_CARD	No relay card installed
34	PI_CNTR_CMD_NOT_ALLOWED_FOR_STAGE	Command not allowed for selected stage(s)
35	PI_CNTR_NO_DIGITAL_INPUT	No digital input installed
36	PI_CNTR_NO_DIGITAL_OUTPUT	No digital output configured
37	PI_CNTR_NO_MCM	No more MCM responses
38	PI_CNTR_INVALID_MCM	No MCM values recorded
39	PI_CNTR_INVALID_CNTR_NUMBER	Controller number invalid
40	PI_CNTR_NO_JOYSTICK_CONNECTED	No joystick configured
41	PI_CNTR_INVALID_EGE_AXIS	Invalid axis for electronic gearing, axis can not be slave
42	PI_CNTR_SLAVE_POSITION_OUT_OF_RANGE	Position of slave axis is out of range
43	PI_CNTR_COMMAND_EGE_SLAVE	Slave axis cannot be commanded directly when electronic gearing is enabled

44	PI_CNTR_JOYSTICK_CALIBRATION_FAILED	Calibration of joystick failed
45	PI_CNTR_REFERENCING_FAILED	Referencing failed
46	PI_CNTR_OPM_MISSING	OPM (Optical Power Meter) missing
47	PI_CNTR_OPM_NOT_INITIALIZED	OPM (Optical Power Meter) not initialized or cannot be initialized
48	PI_CNTR_OPM_COM_ERROR	OPM (Optical Power Meter) Communication Error
49	PI_CNTR_MOVE_TO_LIMIT_SWITCH_FAILED	Move to limit switch failed
50	PI_CNTR_REF_WITH_REF_DISABLED	Attempt to reference axis with referencing disabled
51	PI_CNTR_AXIS_UNDER_JOYSTICK_CONTROL	Selected axis is controlled by joystick
52	PI_CNTR_COMMUNICATION_ERROR	Controller detected communication error
53	PI_CNTR_DYNAMIC_MOVE_IN_PROCESS	MOV! motion still in progress
54	PI_CNTR_UNKNOWN_PARAMETER	Unknown parameter
55	PI_CNTR_NO_REP_RECORDED	No commands were recorded with REP
56	PI_CNTR_INVALID_PASSWORD	Password invalid
57	PI_CNTR_INVALID_RECORDER_CHAN	Data Record Table does not exist
58	PI_CNTR_INVALID_RECORDER_SRC_OPT	Source does not exist; number too low or too high
59	PI_CNTR_INVALID_RECORDER_SRC_CHAN	Source Record Table number too low or too high
60	PI_CNTR_PARAM_PROTECTION	Protected Param: current Command Level (CCL) too low
61	PI_CNTR_AUTOZERO_RUNNING	Command execution not possible while Autozero is running
62	PI_CNTR_NO_LINEAR_AXIS	Autozero requires at least one linear axis
63	PI_CNTR_INIT_RUNNING	Initialization still in progress
64	PI_CNTR_READ_ONLY_PARAMETER	Parameter is read-only

65	PI_CNTR_PAM_NOT_FOUND	Parameter not found in non-volatile memory
66	PI_CNTR_VOL_OUT_OF_LIMITS	Voltage out of limits
67	PI_CNTR_WAVE_TOO_LARGE	Not enough memory available for requested wave curve
68	PI_CNTR_NOT_ENOUGH_DDL_MEMORY	Not enough memory available for DDL table; DDL can not be started
69	PI_CNTR_DDL_TIME_DELAY_TOO_LARGE	Time delay larger than DDL table; DDL can not be started
70	PI_CNTR_DIFFERENT_ARRAY_LENGTH	The requested arrays have different lengths; query them separately
71	PI_CNTR_GEN_SINGLE_MODE_RESTART	Attempt to restart the generator while it is running in single step mode
72	PI_CNTR_ANALOG_TARGET_ACTIVE	Motion commands and wave generator activation are not allowed when analog target is active
73	PI_CNTR_WAVE_GENERATOR_ACTIVE	Motion commands are not allowed when wave generator output is active; use WGO to disable generator output
74	PI_CNTR_AUTOZERO_DISABLED	No sensor channel or no piezo channel connected to selected axis (sensor and piezo matrix)
75	PI_CNTR_NO_WAVE_SELECTED	Generator started (WGO) without having selected a wave table (WSL).
76	PI_CNTR_IF_BUFFER_OVERRUN	Interface buffer did overrun and command couldn't be received correctly
77	PI_CNTR_NOT_ENOUGH_RECORDED_DATA	Data Record Table does not hold enough recorded data
78	PI_CNTR_TABLE_DEACTIVATED	Data Record Table is not configured for recording

79	PI_CNTR_OPENLOOP_VALUE_SET_WHEN_SERVO_ON	Open-loop commands (SVA, SVR) are not allowed when servo is on
80	PI_CNTR_RAM_ERROR	Hardware error affecting RAM
81	PI_CNTR_MACRO_UNKNOWN_COMMAND	Not macro command
82	PI_CNTR_MACRO_PC_ERROR	Macro counter out of range
83	PI_CNTR_JOYSTICK_ACTIVE	Joystick is active
84	PI_CNTR_MOTOR_IS_OFF	Motor is off
85	PI_CNTR_ONLY_IN_MACRO	Macro-only command
86	PI_CNTR_JOYSTICK_UNKNOWN_AXIS	Invalid joystick axis
87	PI_CNTR_JOYSTICK_UNKNOWN_ID	Joystick unknown
88	PI_CNTR_REF_MODE_IS_ON	Move without referenced stage
89	PI_CNTR_NOT_ALLOWED_IN_CURRENT_MOTION_MODE	Command not allowed in current motion mode
90	PI_CNTR_DIO_AND_TRACING_NOT_POSSIBLE	No tracing possible while digital IOs are used on this HW revision. Reconnect to switch operation mode.
91	PI_CNTR_COLLISION	Move not possible, would cause collision
92	PI_CNTR_SLAVE_NOT_FAST_ENOUGH	Stage is not capable of following the master. Check the gear ratio(SRA).
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.
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
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
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_MACRO_DELETE_ERROR	Deleting macro failed
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.
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
2000	PI_CNTR_ALREADY_HAS_SERIAL_NUMBER	Controller already has a serial number
4000	PI_CNTR_SECTOR_ERASE_FAILED	Sector erase failed
4001	PI_CNTR_FLASH_PROGRAM_FAILED	Flash program failed
4002	PI_CNTR_FLASH_READ_FAILED	Flash read failed
4003	PI_CNTR_HW_MATCHCODE_ERROR	HW match code missing/invalid
4004	PI_CNTR_FW_MATCHCODE_ERROR	FW match code missing/invalid
4005	PI_CNTR_HW_VERSION_ERROR	HW version missing/invalid
4006	PI_CNTR_FW_VERSION_ERROR	FW version missing/invalid
4007	PI_CNTR_FW_UPDATE_ERROR	FW update failed
4008	PI_CNTR_FW_CRC_PAR_ERROR	FW Parameter CRC wrong
4009	PI_CNTR_FW_CRC_FW_ERROR	FW CRC wrong
5000	PI_CNTR_INVALID_PCC_SCAN_DATA	PicoCompensation scan data is not valid
5001	PI_CNTR_PCC_SCAN_RUNNING	PicoCompensation is running, some actions can not be executed during scanning/recording
5002	PI_CNTR_INVALID_PCC_AXIS	Given axis can not be defined as PPC axis
5003	PI_CNTR_PCC_SCAN_OUT_OF_RANGE	Defined scan area is larger than the travel range
5004	PI_CNTR_PCC_TYPE_NOT_EXISTING	Given PicoCompensation type is not defined
5005	PI_CNTR_PCC_PAM_ERROR	PicoCompensation parameter error
5006	PI_CNTR_PCC_TABLE_ARRAY_TOO_LARGE	PicoCompensation table is larger than maximum table length

5100	PI_CNTR_NEXLINE_ERROR	Common error in Nexline firmware module
5101	PI_CNTR_CHANNEL_ALREADY_USED	Output channel for Nexline can not be redefined for other usage
5102	PI_CNTR_NEXLINE_TABLE_TOO_SMALL	Memory for Nexline signals is too small
5103	PI_CNTR_RNP_WITH_SERVO_ON	RNP can not be executed if axis is in closed loop
5104	PI_CNTR_RNP_NEEDED	Relax procedure (RNP) needed
5200	PI_CNTR_AXIS_NOT_CONFIGURED	Axis must be configured for this action

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 transfered 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 transfering 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_EXT_PROFILE_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.

10 Troubleshooting

Communication with controller does not work

Communication cable is wrong or defective

⇒ Check cable. Does it work properly with another device?

For RS-232, a null-modem cable must be used.

The interface is not configured correctly

⇒ With the RS-232 interface, check port and baud rate (depending on your controller, the baud rate can be set via DIP switches on the front panel or via a controller parameter). It is recommended that the host PC have a "genuine" RS-232 interface on board. If the host PC uses a USB-to-serial adapter instead, data loss could occur during communication, especially when transferring large amounts of data.

⇒ The USB drivers will make the USB interface appear to all software on the host PC as a new COM port. That port will be present only when the controller is connected via USB and powered on.

⇒ All controllers in a daisy chain network must be set to the same baud rate.

⇒ Up to 16 C-663 controllers can be controlled from a single host computer interface. The RS-232 output stages of some PCs may not be capable of driving more than 6 units; if this is a problem use USB to interface with the PC.

⇒ The RS-232 cable must never be connected to a PC at the same time an USB cable is connected.

Another program is using the interface

⇒ Close the other program.

Specific software has problems

⇒ See if the system works with some other software, e.g. a terminal or development environment. You can, for example, test the communication by starting a terminal program, e.g. PI Terminal, and entering commands like *IDN? or HLP?. Note that multi-character commands are transferred as terminated by a LF (line feed) character and are executed only after the LF is received.

Stage does not move

Cable not connected properly

⇒ Check the connecting cable(s).

Stage or stage cable is defective

⇒ Exchange stage with a working stage to test a new combination of controller and stage.

Controller address wrong or missing

⇒ Check the current controller address (see "DIP Switch Settings" (p. 21)).

⇒ In principle, the address of the target controller is required in every command line, even when recording macros or sending single-character commands. It can only be omitted if the target C-663 has the address 1. See "Target and Sender Address" (p. 75) for more information.

Wrong command or wrong syntax

⇒ Check the error code with the ERR? command (p. 102). "Error Codes" (p. 162) gives the complete error reference.

Wrong axis commanded

⇒ Check if the correct axis identifier is used and if the commanded axis is that of the desired stage (axis identifier also required with single-axis systems!)

Wrong controller parameter settings

⇒ Check the parameter settings with SPA? (p. 145) and SEP? (p. 140). See "Controller Parameters" (p. 27) for more information.

Stage database cannot be opened, or stage selection in host software is not possible.

An error message is displayed saying that the stage database does not have the correct revision

⇒ To support new hardware (controller or stages), it is necessary to release new revisions of the PIUserStages2.dat and *Prefix*UserStages2.dat files. Although PI aims for highest compatibility, the latest host software may not be able to work with older stage database files. You can check the revision of your stage database files using the PIStageEditor (see the PIStageEditor manual for details).

If your PIStages2.dat file does not have the correct revision, download the latest revision from www.pi.ws. For detailed download and replacement instructions see "Updating PIStages2.dat" (p. 45).

The *Prefix*UserStages2.dat allows you to create and save your own stages (see "Adding Stages to User DAT Files" (p. 65)). This database is created the first time you connect stages in the host software (i.e. the first time the VST? or CST functions of the GCS library are used). *Prefix* depends on the GCS library used, e.g. if your controller uses the PI GCS 2 library, *Prefix* will be *PI*. There can be one file of this type for each different GCS library. If you already have a *Prefix*UserStages2.dat file for your controller but this file cannot be opened with the latest software, proceed as follows:

1 Rename the existing *Prefix*UserStages2.dat file on your host PC.

2 Create a new *Prefix*UserStages2.dat. This can be done by opening the stage selection dialog in the host software (e.g. in PIMikroMove) or by calling the VST? or CST functions of the GCS library.

3 Open the new *Prefix*UserStages2.dat in the PIStageEditor.

4 Import the content of the old (renamed) *Prefix*UserStages2.dat file to the new file. See the PIStageEditor manual for details. Note that during the import procedure, the imported stage parameter sets are

converted to fit the new revision. Parameters which were not present in the old revision are set to default values which may need to be optimized. See "Controller Parameters" (p. 27) for details.

Custom software accessing PI drivers does not run.

Wrong combination of driver routines/VIs

⇒ Check if the system runs with a terminal program. If so, read the software manual and compare the sample code from the Mercury product CD with your source code to check the necessary driver routines.

11 Customer Service

For inquiries and orders, contact your PI sales engineer or send us an e-mail (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)
- Software version of driver or host software (if present)
- Operating system on host PC (if present)

The latest versions of the relevant user manuals for your system are available for download on our website (www.pi.ws).

12 Old Equipment Disposal

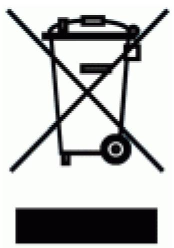
Since 13 August 2005, in accordance with the EU directive 2002/96/EC (WEEE), electrical and electronic equipment can no longer be disposed of in the member states of the EU with other wastes.

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



13 Technical Data

13.1 Specifications

Model	C-663.11
Function	Stepper motor controller, stand-alone capability
Drive type	2-phase stepper motor
Channels	1
Motion and control	
Trajectory profile modes	Trapezoidal, point-to-point
Microstep resolution	1/16 full step
Limit switches	2 x TTL
Reference switches	1 x TTL
Motor brake	1 x TTL
Electrical properties	
Operating voltage	15 to 30 V
Current limitation per motor phase	1000 mA
Interfaces and operation	
Communication interfaces	USB, RS-232 (bus architecture)
Motor connector	Sub-D 15 (f)
Controller network	Up to 16 units* on single interface
I/O ports	4 analog/digital in, 4 digital out
Command set	PI General Command Set (GCS)
User software	PIMikroMove
Software drivers	LabVIEW drivers
Supported functionality	Start-up macro, data recorder
Manual control	Joystick, Y-cable for 2D-motion, pushbutton box
Miscellaneous	
Operating temperature range	0 to 50 °C
Mass	0.3 kg
Dimensions	130 x 76 x 40 mm

*16 with USB; 6 with RS-232 (depending on RS-232 output driver of the PC)

The cycle time of the C-663 is 50 µs. It determines the trajectory update rate and the controller-loop calculation rate.

13.2 Mounting Hole Pattern

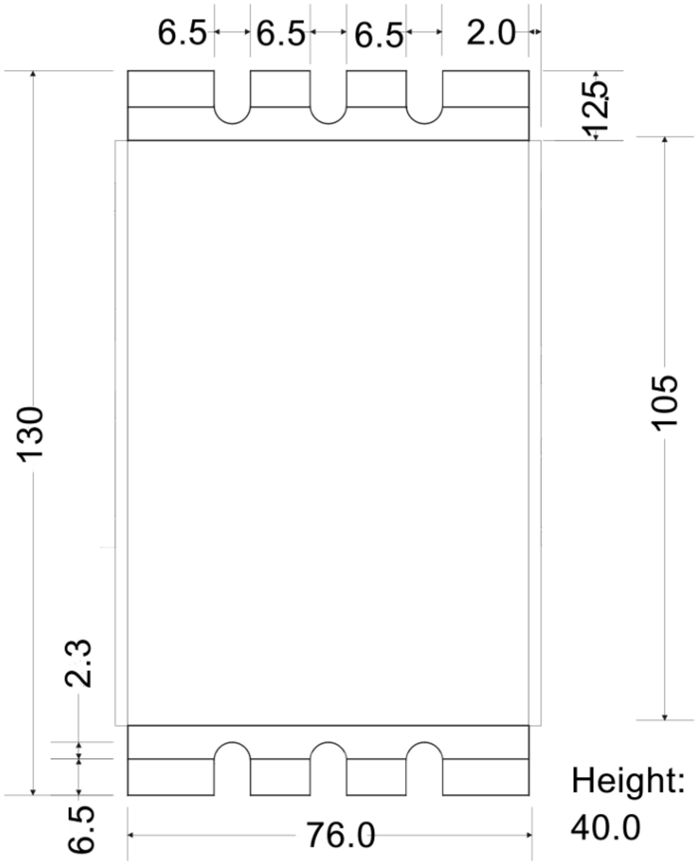
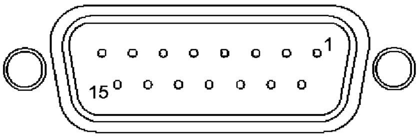


Figure 10: C-663 dimensions in millimeters

13.3 Pin Assignments

13.3.1 Motor Connector

Connector type: Sub-D 15 (f)



CAUTION

Stepper motor drives use the same connector as DC-motor stages but are not compatible. Permanent damage can occur if stage and controller types are not compatible.

The motor control signals are all on the 15-pin sub-D “Stepper Motor only” connector. This connector is compatible with all PI stages having stepper motors. Stages with detachable cables come with the C-815.38 motor cable for connection to the controller.

Pin	Signal direction	Function
1	output	Motor phase 1A
9	output	Motor phase 1B
2	output	Motor phase 2A
10	output	Motor phase 2B
3	input	not used (optional enc. A)*
11	input	not used (optional enc. A-)*
4	input	not used (optional enc. B)*
12	input	not used (optional enc. B-)*
5		n.c.
13	output	Brake signal, active low
6	output	+5V
14	input	Limit, positive, active low
7		GND
15	input	Position reference
8	input	Limit, negative, active low

*With the current firmware, the signals of an external encoder can be recorded with DRC (record option 74) but they cannot be used for position control.

13.3.2 RS-232 In and RS-232 Out Sockets

Connector labels: RS-232 IN and RS-232 OUT
Connector types: Sub-D, 9-pin, male for OUT, female for IN

CAUTION

Never connect the RS-232-IN and USB connectors of the same controller to a PC at the same time as this can cause damage to the controller.

!

The IN and OUT lines are permanently bussed together, straight-through. Only when the USB interface is active, does the controller assert signals on the RxD (receive) line, pin 2, otherwise that line is driven by the host PC only. This is the reason why an RS-232-only network may be limited to as few as 6 units.

Note that the controller RS-232 bus is wired as a DTE and, if connected to a PC, requires a cross-over (null-modem) cable.

Pin on all C-663 Connectors	Signal name on all C-663 Connectors*	Signal direction	Function
1			n.c.
2	RxD*	PC to controller**	Commands
3	TxD*	Controller** to PC	Reports (responses)
4			n.c.
5	GND		GND
6			n.c.
7			n.c.
8			n.c.
9			n.c.

*The RS-232 connection with the PC is via null-modem cable, so the connected signal names on the PC side are reversed.

**If the PC connection is via USB, then the C-663 connected to the PC copies everything received from the host over USB to the C-663 RxD line of *both* its RS-232 connectors. It also copies everything it sees on the C-663 TxD line to the host via USB.
USB Socket
Connector type: industry-standard USB Mini-B

When the USB interface is active, the controller asserts signals on the transmit line (TxD) of the RS-232 connectors for networking up to 15 further controllers. As a result the RS-232 cable must not be connected to the host PC when using USB.

13.3.3 I/O Socket

Connector type: Mini DIN 9-pin

Pin	Signal	Function	Identifier to use in GCS Commands (see below)
1	input	Input 1 (analog: 0 to +5V / digital: TTL)	1
2	input	Input 2 (analog: 0 to +5V / digital: TTL)	2
3	input	Input 3 (analog: 0 to +5V / digital: TTL)	3
4	input	Input 4 (analog: 0 to +5V / digital: TTL)	4
5	output	Output 1 (digital, TTL)	1
6	output	Output 2 (digital, TTL)	2
7	output	Output 3 (digital, TTL)	3
8	output	Output 4 (digital, TTL)	4
9	output	Vcc (+5V)	-

If the Input 1 to Input 4 lines are used as analog input lines:
Their number is reported by the TAC? command (p. 150), their analog input values can be queried with the TAV? command (p. 151) and recorded using the record option 81 of the DRC command (p. 98).

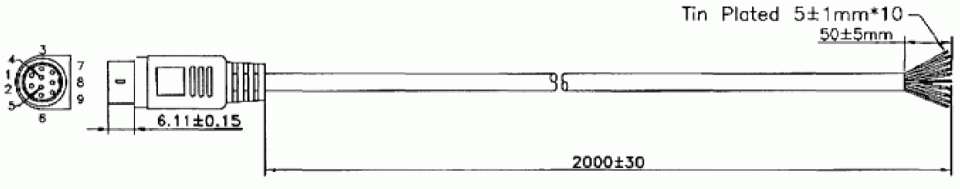
If the Input 1 to Input 4 lines are used as digital input lines:
Their states (high/low) can be queried with the DIO? command (p. 97), the #4 command (p. 82) and the SRG? command (p. 146).

You can use the digital input lines as source of the reference signal, the negative limit signal or the positive limit signal. See parameters 0x5C (p. 31), 0x5D (p. 32) and 0x5E (p. 33) for details.

The states of the Output 1 to Output 4 lines (high/low) can be set using the DIO command (p. 96). Furthermore, you can program the Output 1 to Output 4 lines using the CTO command (p. 90) (trigger configuration) and the TRO command (p. 153) (trigger enabling/disabling). See "Configuring Trigger Output" for examples.

13.3.4 C-170.IO Cable

The C-170.IO cable with open end is equipped with a Mini DIN 9-pin connector. You can use it to make the signals of the I/O socket (p. 191) available. This cable has to be ordered separately.



Specifications:

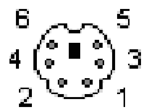
Temperature range: -25 °C to +85 °C
Current rating: 1 A AC/DC
Insulation resistance: 50 MΩ min.
Voltage rating: 50 V AC/DC
Withstanding voltage: 500 V AC for 1 minute

Pin of Mini DIN 9-pin	Wire Color	Signal on I/O Socket of C-663
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)
Shell	Shield, black coated (thicker than the black wire connected to pin 1)	GND

13.3.5 Joystick Socket

Connector type: Mini DIN 6-pin

Pin	Signal direction	Function	Identifier to use in GCS Commands (see below)
1		GND	-
2		n.c.	-
3	output	Vcc (3.3 V)	-
4	input	Input: Joystick Axis analog 0 to 3.3 V	1 for JAS? (p. 114), JAX (p. 115), JAX? (p. 115), JDT (p. 116), JLT (p. 117), JLT? (p. 118) 5 for DRC (p. 98)
5		n.c.	-
6	input	Input: Joystick Button #1	1 for JBS? (p. 116) 6 for DRC

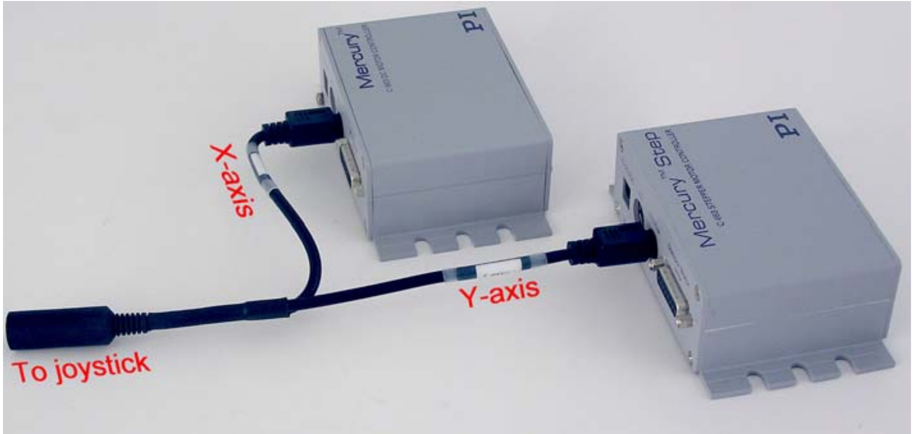


The input values of the joystick axis and the joystick button can be recorded using the record option 81 of the DRC command.

13.3.6 Joystick Y-Cable

The joystick Y-cable (C-819.20Y) maps the signals of the second joystick axis (Y) and button to the first-axis inputs for the second controller. This cable has to be ordered separately.

Joystick Pin	Signal	Controller 1 Pin	Controller 2 Pin
1	GND	1	1
2	Button 2; 0 or 3.3 V		6
3	Vcc (3.3 V)	Vcc (3.3 V)	n.c.
4	X-axis signal, 0 to 3.3 V	4	
5	Y-axis signal, 0 to 3.3 V		4
6	Button 1; 0 or 3.3 V	6	



13.3.7 15-30 VDC Socket

Connector type: barrel connector

Pin	Function	
Center	+15-30 VDC