MS 100E User Manual

C-848 Multi-Axis DC-Motor Controller

Release: 1.1.0 Date: 2007-05-24

This document describes the following product(s):

- C-848.43
  Multi-Axis DC-Motor Controller, 4 Axes
- C-848.23
  Multi-Axis DC-Motor Controller, 2 Axes
- C-848.43i
  Multi-Axis DC-Motor Controller with GPIB interface, 4 Axes
- C-848.23i
  Multi-Axis DC-Motor Controller, with GPIB interface, 2 Axes

© Physik Instrumente (PI) GmbH & Co. KG

Auf der Römerstr. 1 · 76228 Karlsruhe, Germany
Tel. +49 721 4846-0 · Fax: +49 721 4846-299
info@pi.ws · www.pi.ws

Declaration of Conformity

according to ISO / IEC Guide 22 and EN 45014

Manufacturer:	Physik Instrumente (PI) GmbH & Co. KG	
Manufacturer's Address:	Auf der Römerstrasse 1 D-76228 Karlsruhe, Germany	

The manufacturer hereby declares that the product

Product Name:	**Multi-Axis DC-Motor Controller**
Model Numbers:	**C-848**
Product Options:	**all**

complies with the following European directives:

73/23/EEC, Low Voltage Directive

89/336/EEC, EMC Directive

The applied standards certifying the conformity are listed below.

Electromagnetic Emission:	EN 61000-6-3, EN 55011
Electromagnetic Immunity:	EN 61000-6-1
Safety (Low Voltage Directive):	EN 61010-1

4 April 2007
Karlsruhe, Germany

Dr. Karl Spanner
President

Physik Instrumente (PI) GmbH & Co. KG is the owner of the following company names and trademarks:
PILine®
PIMikroMove®

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

Copyright 1999–2007 by Physik Instrumente (PI) GmbH & Co. KG, Karlsruhe, Germany. The text, photographs and drawings in this manual enjoy copyright protection. 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.

Document Number MS 100E, Release 1.1.0
C848_User_MS100E.doc
Eco

This manual has been provided for information only and product specifications are subject to change without notice.

About This Document

Users of This Manual

This User Manual is designed to help the reader to install and operate a motion system which the C-848 Multi-Axis DC-Motor Controller is part of. This manual assumes that the reader has a fundamental understanding of basic servo systems, as well as motion control concepts and applicable safety procedures. The manual also provides the physical specifications and dimensions of the C-848 Multi-Axis DC-Motor Controller as well as the software and hardware installation procedures required to put the motion system into operation.
The newest release of this document is available for download at www.pi.ws or via email: contact your PI sales engineer or write info@pi.ws

Conventions

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

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.

Related Documents

The software tools which are delivered with the C-848 Multi-Axis DC-Motor Controller are described in their own manuals. Updated releases of all documents are available for download at www.pi.ws or via email: contact your Physik Instrumente sales engineer or write info@pi.ws.

Contents

© Copyright 2007 by Physik Instrumente (PI) GmbH & Co. KG

Release: 1.1.0
File:C848_User_MS100E.doc, 633344 Bytes

0.1 Safety Precautions

CAUTION

All motions of the connected motors and mechanical stages are software controlled, and software may fail. Be aware that motorized stages may generate large forces that may cause personal injury or other damage if improperly handled.

This device is intended for use by qualified personnel who are familiar with the safety precautions required to avoid possible injury. Read the manual carefully before operating the device.

If reference mode is switched off stages can be driven into the mechanical hard stop if commanded to a position which is in fact outside the travel range. This can occur with relative moves (MVR) or, if the actual position was incorrectly set (by POS), with absolute moves (MOV)!

1 Introduction

The C-848 Multi-Axis DC-Motor Controller is a multi-channel motion controller based on a multi-processor architecture including multiple fast DSP motion-control chipsets for digital position servo-control, and a separate processor for communications, command parsing, and macro command sequencing. An internal real-time operating system based on a multitasking architecture handles interrupts, I/O operations and user requests.

The C-848 offers high-performance PID motion control with many trajectory-generation and filter-setting options. The connected motors can be controlled as to position, velocity, acceleration and other relevant parameters.

For position feedback, incremental rotary or linear encoders can be used. Many standard PI motors provide resolutions of 2048 counts/rev and are well suited for high-resolution positioning. Even better absolute position accuracy can be achieved with linear incremental scales, available some M-500 and other stage series.

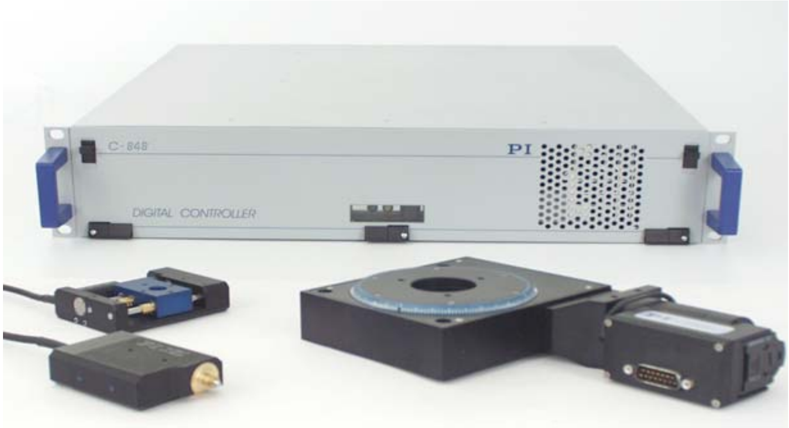


Fig. 1. C-848 *DC-motor controller with a selection of stages*

1.1.1 C-848 Features

- Controls up to 4 DC Motors
- Well-Documented
- RS-232 / IEEE 488 (optional)
- High-Level Command Set
- Macro Command Capability
- Electronic Gearing (see Section 7)

1.2 Applications

Primary applications of the C-848 are found in industrial production and test facilities, as well as in laboratory experimental setups. As a versatile motion and positioning controller for motorized stages, DC-Mike drives and other motorized devices (incl. voice coil) and with its additional digital I/O capabilities, the C-848 offers all the features required for industrial use.

Typical applications are:

- Linear and rotary micropositioning stage motion control
- General laboratory automation
- Production robotics
- Quality inspection automation
- Path tracking
- Laser cutting

1.3 Model Survey

The distinguishing features of the four available models are shown below:

		with IEEE 488 (GPIB) Interface
2 channels	C-848.23	C-848.23i
4 channels	C-848.43	C-848.43i

1.4 Software

With the C-848 DC-Motor Controller all motion of the connected motors and mechanical stages is software controlled.

To meet different application needs, the following software tools are available for download from www.pi.ws. Most are also included on the C-848.CD that comes with the controller:

- *PIMikroMove*® (application for Microsoft Windows platforms) is operating software for this and many other PI controllers. With *PIMikroMove*® you can start your motion system—host PC, controller and stage(s)—immediately without writing customized software. *PIMikroMove*® offers motion control displays and features that in many cases make it unnecessary to deal with ASCII command formats. It also has a complete command input facility, which lets you experiment with various ommands easily. *PIMikroMove*® uses the GCS DLL described below to command the controller (some commands are not available via the current *PIMikroMove* implementation)

- *C-848Control* operating software (GUI for Microsoft™ Windows platforms): *C-848Control* also permits you to start your motion system—host PC, controller and stage(s)—immediately without the need to write customized software. Its command input facility represents an easy way to experiment with all the commands supported by the C-848.

- LabVIEW drivers (support the PI General Command Set (GCS) based on ASCII communication): LabVIEW is a software tool, which must be purchased separately from National Instruments. for data acquisition and process control. The PI LabVIEW software consists of a collection of virtual instruments (VIs) to run the C-848 and control its connected axes.
- GCS.DLL (Windows DLL Library): The C-848 supports a higher level command set—the *PI General Command Set* (*GCS*; GCS.DLL)—which is based on ASCII communication with well-defined commands and replies. Most PI controllers understand this command set and thus your applictions are controller independent.
- COM module (for Microsoft™ Windows platforms with COM support installed), recommended for Visual Basic programmers, similar to GCS.DLL in functionality

Updated releases are available on the download area of www.pi.ws or via email: contact your PI sales engineer or write info@pi.ws.

1.5 Units and GCS

1.5.1 Hardware and Physical Units

The GCS (General Command Set) system uses physical units of measure. Most controllers and GCS software have default conversion factors chosen to convert hardware-dependent units (e.g. encoder counts) into basic physical units like mm or degrees, as appropriate (see SPA and SPA? command descriptions, parameters 14 and 15). The defaults are generally taken from a database of stages used by the operating software (PIStages.dat). An additional scale factor can be applied (see DFF command), making a second physical unit available without overwriting the conversion factor for the first.

1.5.2 Rounding Considerations

When converting move commands in physical units to the hardware-dependent units required by the motion control layers, rounding errors can occur. The GCS software is so designed, that a relative move of x physical units will always result in a relative move of the same number of hardware units. Because of rounding errors, this means, for example, that 2 relative moves of x physical units may differ slightly from one relative move of 2x. When making large numbers of relative moves, especially when moving back and forth, either intersperse absolute moves, or make sure that each relative move in one direction is matched by a relative move of the same size in the other direction

Examples

Assuming 5 hardware units = 33 x 10^{-6} physical units:

Relative moves	**cause move of**
smaller than 0.000003 physical units	0 hardware units
of 0.000004 to 0.000009 physical units	1 hardware unit
of 0.000010 to 0.000016 physical units	2 hardware units
of 0.000017 to 0.000023 physical units	3 hardware units
of 0.000024 to 0.000029 physical units	4 hardware units

Hence:

2 moves of 10 x 10^{-6} physical units followed by 1 move of 20 x 10^{-6} in the other direction cause a net motion of 1 hardware unit forward.

100 moves of 22 x 10^{-6} followed by 200 of -11 x 10^{-6} result in a net motion of -100 hardware units

5000 moves of 2 x 10^{-6} result in no motion

1.6 About This Manual

This manual familiarizes you with the features of the C-848 DC-Motor Controller, tells you how to start the system, introduces macro programming and describes the commands of the PI General Command Set (GCS) which must be used with the C-848.

The software tools mentioned above are described in their own manuals. All documents for download from www.pi.ws or by email: contact your PI Sales Engineer or write info@pi.ws.

2 Preparing for Operation

2.1 Unpacking and Inspection

The C-848 was carefully inspected, both electrically and mechanically, before shipment. Check for any obvious signs of physical damage that may have occurred during transit.

The following items are shipped with every C-848 order:

- ➢ C-848 Controller
- ➢ Line cord
- ➢ RS-232 Null-modem cable (C-815.34)
- ➢ Distribution Software: "C-848 TOOLS"
- ➢ MS 101E User Manual (this document)

2.2 Power Requirements

The C-848 has a wide-range-input power supply that can be used from 90 VAC to 240 VAC at 50, 60 or 440 Hz. Connect the line cord to the AC input on the rear panel of the device.

2.3 Communication

The C-848 is controlled from a host computer (not included) via an RS-232 serial connection or over the optional GPIB (IEEE 488) interface with ASCII commands. It is also possible set up the C-848 for operation without a host computer, having it automatically execute a macro upon startup. The macro can perform a preprogrammed set of commands (e.g. moves).

2.3.1 RS-232 and RS-422 Interfaces

The serial communications ports are accessed via DB 9m connectors on the rear panel of the controller. COM1 follows the RS-232 protocol, while the COM2 port follows the RS-422 (differential) signal protocol. From the software side, the protocols are identical, but RS-422 signals can be used over greater distances than RS-232. To use RS-422, you need an RS-422 port on the host computer, or an adapter box between the host and controller.

The ports are set to the following parameters:

57,600 baud, 8 bits, no parity.

See the software manual if you need to change these values (baud rate change is not currently supported with *PIMikroMove*).

The port settings on the host computer and controller must match for communication to be established, and communication is required for changing the settings on the controller from the host PC.

2.3.2 GPIB Interface

If the optional GPIB (IEEE 488) interface is present, it can be used in place of the RS-232 or RS-422 interface. GPIB is a bus-oriented protocol, so no two devices should have the same address setting. The default settings should allow establishing communications; see the software manual if you need to change them.

3 Quick Start

C848Control offers the most convenient user interface for setting up the system, and it will be used in the Quick Start instructions below (*C848Control* is described in detail in a separate manual).

1. The C-848 is equipped with a wide range power supply that accpepts from 90 - 240 VAC at 50, 60 or 440 Hz. With at least one of the two power switches off, connect the unit to line power.
2. Connect the stages/motors to the C-848.
3. Connect the communications cable between the controller and the host computer. For serial communications use the included null-modem cable and connect it to the connector labeled "RS-232".
4. If you will be using a joystick, plug it into the socket labeled "Joystick" at the rear of the controller. Note that direct connection of a joystick is supported by older models only. *PIMikroMove* and the LabVIEW driver set support a joystick on the host computer.

5. If external triggering/signalling is to be used, connect the external inputs/outputs to the "Digital I/O" socket at the rear of the controller.
6. Turn the controller power on (both front- and rear-panel power switches must be ON). The power LED visible in the window on the front panel will come on and the controller will issue a short beep. The start sequence takes less than 90 seconds and ends with a one-second 440 Hz (A) tone. On power-on, all parameters are set to their power-up default values.
7. Install and start *C-848Control* on the host computer. Consult the *C-848Control* software manual for details.
8. Type the command: "*IDN?" asking for device identification message. The response should be similar to the following:
   `C-848, MultiAxis DC-Motor Controller V1.10 1704030930`
9. Move the axes you want to work with to their references or a known position (for example: REF A or, if Axis A has no reference switch, MNL A).
10. Start a move: type, for example, "MOV A 1.234”
    Axis #1 will then move to position 1.234
11. If you have a directly connected a joystick (older models only) and wish to use it to control motor axes A and B, all you need to do is enable joystick operation with JEN 1 (p. 21). To assign the joystick axes to different motor axes, the SJA command (p. 30) can be used. If axis motion occurs when the joystick is centered, calibrate the joystick with the JEN CAL command (p. 21). See the *PIMikroMove* manual for use of a joystick connected to the host PC.

Notes:

C-848Control is set by default to send an ERR? command after each command: “ERR?” in the *Terminal* window does not indicate an error unless the response in the *Response* window is non-zero! A listing of error codes is available on the *Help* menu.

If the *Terminal* window stays dimmed after motion has ceased, try pressing the blue *Stop* button.

4 Referencing

Because the encoder signals used for position feedback provide only releative motion information, 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, or an absolute position value can be entered by command.

4.1 Referencing Mode

If an axis has referncing mode “OFF”, (see the RON command description), then relative moves are permitted without the axis having been referenced. If referencing mode is “ON” then the reference status determines whether moves are allowed.

4.2 Reference Status

With status “unreferenced” and referencing mode “ON,” no moves except referencing moves are allowed.

NOTE

the REF command requires that the axis have a reference switch, MPL and MNL require that the axis have limit switches.

CAUTION

!

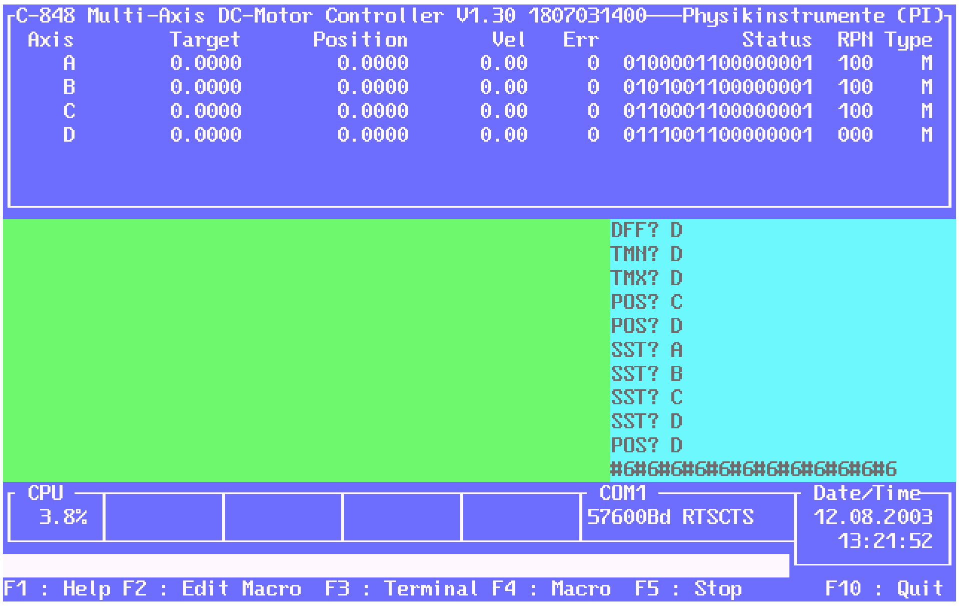
If the controller is given an incorrect position with POS, the axis can be run into a hard stop.

To change the reference status of an axis from “unreferenced” to “referenced.” you can:

- Perform a reference move with REF, MPL, or MNL
- Set referencing mode of OFF with RON and inform the controller of the absolute position of the axis with POS.

5 Controller Keyboard and Monitor

It is possible to connect an external keyboard and VGA monitor to the C-848.



```
C-848 Multi-Axis DC-Motor Controller V1.30 1807031400——Physikinstrumente (PI)
 Axis         Target       Position        Vel    Err           Status   RPN Type
    A         0.0000         0.0000       0.00      0  0100001100000001  100    M
    B         0.0000         0.0000       0.00      0  0101001100000001  100    M
    C         0.0000         0.0000       0.00      0  0110001100000001  100    M
    D         0.0000         0.0000       0.00      0  0111001100000001  000    M

                                                    DFF? D
                                                    TMN? D
                                                    TMX? D
                                                    POS? C
                                                    POS? D
                                                    SST? A
                                                    SST? B
                                                    SST? C
                                                    SST? D
                                                    POS? D
                                                    #6#6#6#6#6#6#6#6#6#6#6#6
 CPU                                                COM1               Date/Time
  3.8%                                              57600Bd RTSCTS      12.08.2003
                                                                          13:21:52
F1 : Help F2 : Edit Macro  F3 : Terminal F4 : Macro  F5 : Stop        F10 : Quit
```

Fig. 2 Screenshot of optional monitor connected to controller, shown shortly after establishing connection with host

The monitor shows all commands as they are entered as well as other information, such as axis positions and position error.

With the function keys listed in the menu bar at the bottom of the screen, it is possible to enter commands directly.

6 External Triggering/Signalling

It is possible to trigger external devices, and receive signals from external devices, using the digital IO lines. See the DIO and DIO? commands for programming details and the Digital I/O Connector Section p. 45 for signal locations.

7 Electronic Gearing

The C-848 offers an electronic gearing feature. Using the SRA (p. 32) and EGE (p. 18) commands, it is possible to link a “master” and a “geared” axis, so that motion of one automatically entails proportional motion of the other. See the command descriptions for details.

8 Macro Programming

This feature allows defining command sequences and storing them permanently in non-volatile memory. Each macro command 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 controller is started, facilitating stand-alone operation without a host computer.

8.1 Macro Definition

To define a macro command sequence, first activate macro programming mode with the command: **"MAC BEG** *name***"**, where *name* is a user-settable name with a maximum of 8 characters.

While in programming mode, commands are not executed but stored in macro storage.

Programming mode is exited by the **"MAC END"** command.

Example (note how *macro3* calls macros #1 and #2 for execution):

```
MAC BEG macro1
MVR A12.5
WAI A
END

MAC BEG macro2
MVR A-12.5
WAI A
MAC END

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

While in programming mode, the following command cannot be used:

MAC BEG	define a macro

8.2 Starting Macros

A defined macro can be run by the command: MAC START *name* where *name* is the name that was given to the macro to be run.

8.3 Predefined Macro Names

If a macro with the name STARTMAC exists, it is executed automatically every time the C-848 is started.

In this macro you can store all necessary initialization commands, e.g. SST, SCA, and so on.

If a macro with the name C848MAC exists, it can be executed with a single keystroke by pressing the F4 key on an external keyboard plugged into the keyboard socket of the controller. This can be used for demo purposes.

9 Commands

Commands are used to set operating modes, transfer motion parameters 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.

9.1 Overlapping Commands

Overlapping commands are those which require a certain time for execution. For example, move commands need some time until the desired target is reached and the command is completed. Any new move command would reset the target to a new value and the old one would never be reached.

To avoid this, a *command-in-execution* or *pending* flag is set during the execution of an overlapping command. Overlapping commands can be sequenced by using the WAI command. If any command in execution has the pending flag set for the specified axis, the WAI command holds the next command until the movement has finished. This allows sending the next move command before the currently set target has been reached without aborting the current move, as shown in the examples below:

Commands	Action
MOV A25 MOV A15	Axis-A motor starts at 0 and moves to position 25. The previous target of 25 is not reached because it is overwritten by next command
MOV A25 WAI A MOV A15	Axis-A motor starts at 0 and moves to position 25. The **“WAI A”** waits until axis A has finished the move before the following command is executed. After the position 25 has been reached, the motor moves back to position 15.

9.2 Sequential Commands

Commands which do not set the pending flag are “sequential commands.” Sequential commands are executed and terminated immediately and need no *time* for execution. For example, setting a parameter to a new value is executed at once.

9.3 Command Format

Commands are transmitted as ASCII characters and have the format below: Exceptions are the single character binary commands on p. 39 ff.

CMDSPXsV.V[XsV.V]…LF

where:

CMD token (mneumonic) of the specific command
SP one space (ASCII char #32),
X axis identifier (A , B , C, ...),
s sign (positive values can be transmitted without sign)
V.V parameter, values are doubles (double precision) or integers, depending on the command.
[…] Square brackets “[ ]” indicate an optional entry or parameter. Note that it is possible to access more than one channel (axis) in one command line.

$\boxed{\text{LF}}$ LineFeed (Char #10).

Example:

Send: MOV$\boxed{\text{SP}}$A10.0B5.0C20.0$\boxed{\text{LF}}$

Moves axis A to position 10.0 mm, axis B to 5.0 mm and axis C to 20.0 mm

Format of answers:

Some commands deliver a report message having the following format:

X=sV.V$\boxed{\text{LF}}$

where:

X axis identifier
s sign (positive values are transmitted without sign)
V.V parameter, values are doubles or integers depending on the command
$\boxed{\text{LF}}$ LineFeed (ASCII char #10).

Example:

Send: POS?$\boxed{\text{SP}}$AB$\boxed{\text{LF}}$

Report: A=10.0000$\boxed{\text{SP}}$ $\boxed{\text{LF}}$
B=5.0000$\boxed{\text{LF}}$

There is one space (char #32) before the LineFeed character on all lines of the response *except* the last line.

The individual spaces and linefeed characters will not all be marked in the rest of this manual.

Floating Point Data Format

Some commands require parameters in floating point format. The following syntax is possible for these parameters:

sv
sv.v
sv.vEsxxx

where:

s sign(positive values can be without sign)
v integer parameter, will be converted into float by firmware
v.v float parameter, the decimal separator must be ".", not ","
E exponent character
xxx exponent value

The format in which floating point values are reported (output) is always:

sv.vvvv

where:

s sign (positive values are reported without sign)
v.vvvvvv always with 4 digits after decimal point

if the reply includes more than 2 floats, each will occupy one line.

9.4 Command List

*IDN? (Get Identity Number)
BRA? (get axes with BRAke)
BRA (BRAke)
CLR (CLeaR axis status)
CLS (CLear controller Screen)
CST (Change STage)
CST? (get stagename)
DEL (DELay)
DEM (DEMo)
DFF (DeFine Factor)
DFF? (get factor)
DFH (DeFine Home
DFH? (get home positions)
DIO (set Digital Output)
DIO? (get Digital Output)
DSP (DiSPlay)
DSP? (get displayed axes)
EGE (Electronic Gearing)
EGE? (get electronic gearing)
ERR? (get ERRor)
GOH (GO Home)
HID (HIDe)
HID?
HLP? (HeLP)
HLT (HaLT)
INI (INItialization)
ITD (Init To Default)
JEN (Joystick Enable)*
JEN? (get joystick status)*
LIM?
MAC BEG (start recording macro)
MAC DEL (delete macro)
MAC END ()
MAC START (start MACro)
MAC? (list macro)
MAS? (list master axis)
MNL (Move to Negative Limit)
MOV (MOVe absolute)
MOV? (read target position)
MPL (Move to Positive Limit)
MSG (display MeSsaGe)
MVR (MoVe relatiVe)
NLM (set Negative soft LiMit)
NLM? (get negative soft limit)
ONT? (axis ON Target)
PLM (set Positive soft LiMit)
PLM? (get positive soft limit)
POS (set actual position)
POS? (read real POSition)

REF (move to REFerence position)
REF?
RON (set Reference mode ON | off)
RON? (get Reference state)
RST (ReSTore parameter)
SAI (Set Axis Identifier)
SAI? (get axis identifier)
SAV (SAVe axis parameters)
SCA (Set Cursor controlled Axes)
SCA? (get cursor controlled axes)
SJA (Set Joystick-controlled Axes)*
SJA? (get Joystick-controlled axes)*
SMO (Set Motor Output)
SMO? (Get Motor Output)
SPA (Set Parameter)
SPA? (Get Parameter)
SRA (Set Ratio)
SRA? (get ratio)
SSL (Set Soft limit on or off)
SSL? (Get Axes with Soft Limit on)
SSN? (Get serial Number)
SST (Set Step size)
SST? (Get Step size)
STE (STEp function)
STA? (Get Status)
STE?
STP (Stop Motion)
SVO (set SerVO on or off)
SVO? (Get servo status)
TIM? (Tell System Time)
TIO? (Tell Digital I/O s)
TMN? (Tell Minimum Travel Value)
TMX? (Tell Maximum Travel Value)
TNJ? (Tell number of Joysticks)
TVI? (Tell Valid axis Identifiers)
VEL (Set Velocity)
VEL? (Get Velocity)
VER? (Get Version)
VMO (Virtual MOtion)
VST? (Get available Stages)
WAA (Wait for completion on all axes)
WAI (Wait for completion)
#5 (Poll the Motion Status)
#6 (Position Change?)
#7 (Controller Ready?)
#8 (Macro running?)
#24 (Stop all axes)
#27 (ESC) (System Abort)

* Directly connected joystick is supported by older models only. See the *PIMikroMove* manual for support of a joystick on the host computer.

9.5 Command Reference

*IDN? (Get Identity Number)

Description: Reports the device identity number.

Format: *IDN?

Parameters: none

Response: One-line string terminated by line feed, e.g.:
C-848,Multi-Axis Motion Controller V1.10 1704030930

BRA? (get axes with BRAke)

Description: Lists the axes with brakes.

Format: BRA?

Parameters: none

Response: identifiers of axes with brakes
e.g.: AC
if no axis has a brake, the answer is an empty line.

BRA (BRAke)

Description: Sets brake on or off

Format: BRA A*n*[B*n*][C*n*][D*n*]

Parameters: A,B,C,D axis identifiers, (on C-848.x0, A and B only)
n=1 (or “on”): Set brake on
n=0 (or “off”): Set brake off

Response: none.

Troubleshooting: axis has no brake

CLR (CLeaR axis status)

Description: Clears the axis status.

Format: CLR[A][B][C][D]

Parameters: A,B,C,D ... axis identifiers

Response: none

CLS (CLear controller Screen)

Description: Clears the (optional) VGA monitor on the controller

Format: CLS

Parameters: none

Response: none

CST (Change STage)

Description: Sets the stage type.

Format: CST A[ ]*stagename*[B[ ]*stagename*]...

Parameters: A: axis identifier, *stagename*: name of the stage to select
The known stages can be listed with the VST? command

Response: 1: if selection ok,
0: if *stagename* not found

Troubleshooting: Unknown stage

CST? (get stagename)

Description: Replies with the name of the stage.

Format: CST?[A][B][C] ...

Parameters: A ,B,C: axis identifiers

Response: A=stagename1
B=stagename2
C=stagename3

DEL (DELay)

Description: Delays x milliseconds

Format: DEL *v.vv*

Parameters: *v.vv* time to delay [millisec]

Response: none

DEM (DEMo)

Description: Moves all connected axes randomly,
Motion can be stopped with F5 on the controller keyboard or with the STP command

Format: DEM

Parameters: none

Response: none

DFF (DeFine Factor)

Description: Scale factor for physical units, e.g. a factor of 25.4 sets the physical units to inches. Changing the scale factor will change the numerical results of other commands, but not the underlying physical magnitudes.

Format: DFF A*v.vv*[B*v.vv*][C*v.vv*]...

Parameters: A,B,C,D: axis identifier, *v.vv* value to set

Response: none

Troubleshooting: Illegal axis identifier

DFF? (get factor)

Description: Gets the scale factor set by the DFF command

Format: DFF?[A][B][C][D]

Parameters: A,B,C,D: axis identifier

Response: scale displacement values of the requested axis

Troubleshooting: Illegal axis identifier

DFH (DeFine Home

Description: Makes the current position the new home position.

Format: DFH[A][B][C][D]

Parameters: A,B,C,D: axis identifier

Response: none

Troubleshooting: Illegal axis identifier

DFH? (get home positions)

Description: Gets home position (absolute)

Format: DFH?[A][B][C][D]

Parameters: A,B,C,D: axis identifier

Response: none

Troubleshooting: Illegal axis identifier

DIO (set Digital I/O)

Description: Switches the specified output line(s) to specified state(s). Can be used to trigger external devices.

Format: DIO[A0|1][B0|1][C0|1]...

Parameters: A,B,C,...,H: output line specification.

0 for LOW , 1 for HIGH

Response: none

DIO?

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

Format: DIO?[A][B][C][D]

Parameters: A,B,C,...,H: digital input line specification.

Response: A=1: if input line is high
A=0: if input line is low

DSP (DiSPlay)

Description:	Displays the specified axis (hidden by the HID command) on the controller screen
Format:	DSP[A][B][C][D]
Parameters:	A,B,C,D: axis identifier
Response:	none
Troubleshooting:	Illegal axis identifier

DSP? (get displayed axes)

Description:	Lists the axes currently displayed on the controller screen.
Format:	DSP?
Parameters:	none
Response:	identifiers of the currently displayed axes

EGE (Electronic Gearing)

Description:	Activates electronic gearing for the specified axes
Format:	EGE A0|1 [B0|1]
Parameters:	A,B: axis identifier. 0 for OFF, 1 for on
Response:	none
Troubleshooting:	illegal axis for electronic gearing

EGE?

Description:	Lists the state of electronic gearing for the specified axes
Format:	EGE? [A[B]]
Parameters:	A,B: axis identifier.

Response:	A=1: electronic gearing on
	A=0: electronic gearing off

ERR? (get ERRor)

Description: Get Error code (Error code refers to the previously transferred command).

Format: ERR?

Parameters: none

Response: Error number

Troubleshooting: RS-232 communication breakdown

Error codes

0	No error
1	Parameter syntax error
2	Unknown command
5	MOV of unreferenced stage or servo off
6	Reserved
7	Motion range exceeded.
8	Velocity range exceeded.
9	Reserved
10	Stop was executed
11	SST parameter out of range
12	Reserved
13	Reserved
14	Reserved
15	Wrong axis identifier
16	Unknown stage name
17	Parameter out of range
18	Invalid macro name
19	Error recording macro
20	Macro not found
21	Axis has no brake
22	Duplicate axis identifier
23	Illegal axis identifier
24	Illegal parameter number
25	Illegal floating-point number
26	Parameter missing
27	Soft limit exceeds hard limit
28	Reserved
29	No more step response values
30	No step response values recorded
32	Stage has no limitswitches
33	Reserved
34	Command not allowed for this stage
40	No Joystick connected
41	Illegal Axis for Electronic Gearing
42	Slave Position out of limit
1000	Too many nested macros

GOH (GO Home)

Description: Moves the axis to its home (reference) position

Format: GOH[A][B][C][D]

Parameters: A,B,C,D: axis identifier

Response: none

Troubleshooting: Illegal axis identifier

HID (HIDe)

Description: Hides the specified axes on the controller screen

Format: HID[A][B][C][D]

Parameters: A,B,C,D: axis identifier

Response: none

Troubleshooting: Illegal axis identifier

HID?

Description: lists hidden axes

Format: HID[A][B][C][D]

Parameters: A,B,C,D: axis identifier

Response: none

Troubleshooting: Illegal axis identifier

HLP? (HeLP)

Description: Lists all available commands

Format: HLP?

Parameters: none

Response: Command list

Troubleshooting: RS-232 communication breakdown

HLT (HaLT)

Description: Stops the motion of the specified axes. Any target position can be subsequently reached by the MOV or MVR command, and the POS? command gives the actual position. Does not work during MPL, MNL or REF motion (use #24 instead)

Format: HLT[A][B][C][D]

Parameters: A,B,C,D: axis identifier(s)

Response: none

Troubleshooting: Illegal axis identifier

INI (INItialization)

Description: Initializes motion control chip for the axis, resets soft limits, sets reference state to "not referenced," and if axis is under control of a directly connected joystick (older models only), joystick is disabled.

Format: INI[A][B][C]

Parameters: A, B, C axis identifiers of axes to be initialized.

Response: none

Troubleshooting: Illegal axis identifier

ITD (Init To Default)

Description: Restore axis parameters (including copy saved with SAV command) to factory default values

Format: INI[A][B][C]

Parameters: A, B, C axis identifiers of axes to be restored.

Response: none

Troubleshooting: Illegal axis identifier

JEN (Joystick ENable)

Description: Enables or disables directly connected joystick (older models only). Power-up default is “disabled”.
The axes which are controlled by the joystick (A and B unless changed with SJA) must be referenced. When the joystick is enabled, the joystick axes cannot commanded by MOV or MVR commands.

Format: JEN value

Parameters: *value* : 1 enables, 0 disables the joystick, CAL calibrates the joystick

Response: none if *value* is 0 or 1

if *value* is CAL :

Find Center Point
Leave the handle centered, then press a button

Axis Calibration
Move the handle in complete circles, then press a button

Verify Center Point
Leave the handle centered, then press a button

The calibaration is finished

Troubleshooting: No joystick connected

JEN?

Description: Lists the activation state of the joystick.

Format: JEN?

Parameters: none

Response: 0 : joystick disabled
1 : joystick enabled

LIM?

Description: Indicates whether axes have limit switches

Format: LIM?[A][B][C][D]

Parameters: A,B,C,D: axis identifier

Response: 1: if the axis has limit switches
0: if the axis has no limit switches

Troubleshooting: Illegal axis identifier

MAC BEG (start recording macro)

Description: Enters macro recording mode. In programming mode, commands entered are not executed but stored as a named macro. See also the “MAC END” command.

Format: MAC BEG *macroname*

Parameters: *macroname*: name of the macro to be recorded

Response: none

MAC DEL (delete macro)

Description: Deletes Macro

Format: MAC DEL macroname

Parameters: macroname: Name of macro to be deleted

Response: none

Troubleshooting: Undefined macro name

MAC END ()

Description: Finishes programming of the macro named in the preceeding MAC BEG command

Format: MAC END

Parameters: none

Response: none

Troubleshooting: Not in macro programming mode

MAC START (start MACro)

Description: Starts the specified macro

Format: MAC START *macroname*

Parameters: *macroname*: name of the macro

Response: none

Troubleshooting: Macro *macroname* not found

MAC? (list macro)

Description: Lists the specified macro

Format: MAC [*macroname*]

Parameters: *macroname*: name of the macro; without parameter all stored macros are listet

Response: none

Troubleshooting: Macro *macroname* not found

MAS?

Description: Lists the master axis for the specified axes

Format: MAS? [A[B]]

Parameters: A,B: axis identifier.

Response: A=C
B=D

MNL (Move to Negative Limit)

Description: Moves the axis to the negative limit switch, initializes the motion control chip (including resetting soft limits), sets the position counter to 0, sets the reference state to "reference OK"

Format: MNL[A][B][C]

Parameters: A, B, C axis identifiers

Response: none

Troubleshooting: Illegal axis identifier

MOV (MOVe absolute)

Description: Move to absolute position. A blank is required before each axis specifier.

The values are interpreted as floating point format.

Internal accuracy for all transformations is 18 digits.

The controller checks if the programmed position can be reached before it starts motion. If the target is outside the travel range of the connected stage, the move will not be started and error code 7 is set.

Axes not explicitly specified retain their previous states.

Example: After initialization the system position is:

A=0; B=0; C=0; D=0;

The new command is: MOV B3.245678 A1.23 C4.56789

The new target position is:

A= 1.23 mm; B= 3.2457 mm; C= 4.5679 mm; D=0 mm

After a new command MOV D1.2 the new target positions are:

A= 1.23 mm; B= 3.2457 mm;´C= 4.5679 mm; D= 1.2 mm;

Format: MOV A*v.vv*[B*v.vv*][C*v.vv*]...

A,B,C,D: axis identifiers
v.vv Parameter [physical units]

Response: none

Troubleshooting: Parameter out of limits, Illegal axis identifier, joystick enabled for axis

MOV? (read target position)

Description:	Reports the current commanded position (target) of the given axis, which resulted from a move command.
Format:	MOV?[A][B][C]
Parameters:	A,B,C,D: axis identifier
Response:	Target position of the specified axes
Troubleshooting:	Illegal axis identifier

MPL (Move to Positive Limit)

Description:	Moves the axis to the positive limit switch, initializes the motion control chip (including resetting soft limits), sets the position counter to the maximum travel value and sets the reference state to "reference OK".
Format:	MPL[A][B][C]
Parameters:	A, B, C axis identifiers
Response:	none
Troubleshooting:	Illegal axis identifier

MSG (display MeSsaGe)

Description:	Displays a message on the display of the controller.
Format:	MSG *message_text*
Parameters:	*message_text*: text to display
Response:	none

MVR (MoVe relatiVe)

Description: Move to relative position. A blank is required before each axis identifier.

The values are interpreted as floating point format.

Internal accuracy for all transformations is 18 digits.

The controller checks if the programmed position can be reached before it starts motion. If the target exceeds the travel range of the connected stage, the move will not be started and error code 7 is set.

Axes not specified retain their previous settings.

Example: After initialization the system position is:

A=0; B=0; C=0; D=0;

The new command is: MVR B3.245678 A1.23 C4.56789

The new target position is:

A= 1.23; B= 3.2457; C= 4.5679; D=0

After the new command MVR A-1 the new target positions are:

A= 0.23; B= 3.2457;C= 4.5679; D= 0;

Format: MVR A*v.vv*[B*v.vv*][C*v.vv*]...

A,B,C,D: axis identifiers
v.vv parameter [physical units]

Response: none

Troubleshooting: Parameter out of limits, Illegal axis identifier, joystick enabled for axis

NLM (set Negative soft LiMit)

Description: Sets the negative soft limit for the specified axis.

Format: NLM A*v.vv*[B*v.vv*][C*v.vv*]...

A,B,C,D: axis identifiers
v.vv Parameter [physical units]

Response: none

Troubleshooting: Parameter out of limits or illegal axis identifier

NLM? (get negative soft limit)

Description: Gets the negative soft limit for the specified axis.

Format: NLM?[A][B][C][D]

A,B,C,D: axis identifiers

Response: Soft limits of the specified axes

Troubleshooting: Illegal axis identifier

ONT? (axis ON Target)

Description: Reports whether the specified axis has reached the target position.

Format: ONT? [A][B][C][D]

A,B,C,D: axis identifiers

Response: 1: the axis is on target
0: the axis is not on target

Troubleshooting: Illegal axis identifier

PLM (set Positive soft LiMit)

Description: Sets the positive soft limit for the specified axis.

Format: PLM A*v.vv*[B*v.vv*][C*v.vv*]...

A,B,C,D: axis identifiers
v.vv Parameter [physical units]

Response: none

Troubleshooting: Parameter out of limits or Illegal axis identifier

PLM? (get positive soft limit)

Description: Gets the positive soft limit for the specified axis.

Format: PLM?[A][B][C][D]

A,B,C,D: axis identifiers

Response: Soft limits of the specified axes

Troubleshooting: Illegal axis identifier

POS (set real POSition)

Description: Informs the controller of the actual position of the specified axes, which must have reference mode OFF (does not cause axis to move).

Format: POS[Av.vv]Bv.vv][Cv.vv]...

Parameters: A,B,C,D: axis identifier.
v.vv position value [physical units]

Response: none

Troubleshooting: Reference mode ON,
Illegal axis identifier

POS? (read real POSition)

Description:	Reports the actual position of the specified axes, even if the last move was interrupted or has not been finished.
Format:	POS?[A]B][C]...
Parameters:	A,B,C,D: axis identifier.
Response:	Actual position of the specified axes.
Troubleshooting:	Illegal axis identifier

REF?

Description:	Indicates whether queried axes have a reference switch or not
Format:	REF?[A][B][C][D]
Parameters:	A,B,C,D: axis identifier.
Response:	1: if the axis has reference sensor 0: if the axis has no reference sensor
Troubleshooting:	Illegal axis identifier

REF (move to REFerence position)

Description:	Moves the specified axes to the reference position and sets the reference state to "reference OK".
Format:	REF[A][B][C][D]
Parameters:	A,B,C,D: axis identifier.
Response:	1: if the reference position is reached 0: if the reference position was not reached, or the stage has no reference sensor
Troubleshooting:	Illegal axis identifier

RON

Description: Sets the reference mode
If reference mode is OFF, no referencing is required for the axis. Unless the controller is informed of the actual position of the axis (with POS), only relative moves can be commanded (using MVR). For stages with neither reference nor limit switch, reference mode is always OFF.

Warning

If reference mode is switched off stages can be driven into the mechanical hard stop if commanded to a position which is in fact outside the travel range. This can occur with relative moves (MVR) or, if the actual position was incorrectly set (by POS), with absolute moves (MOV)!

Format: RON [A0|1][B0|1][C0|1]...

Parameters: A,B,C,D: axis identifier.
0 for OFF, 1 for on

Response: none

Troubleshooting: Illegal axis identifier

RON?

Description: Lists the reference state

Format: RON? [A][B][C]...

Parameters: A,B,C,D: axis identifier.

Response: A=1 REF state is on
A=0 REF state is off

Troubleshooting: Illegal axis identifier

RST (ReSTore parameter)

Description: Loads back the stage configuration and the motion parameters which were last saved with SAV.

Format: RST[A][B][C][D]

Parameters: A,B,C,D: axis identifier.

Response: none

Troubleshooting: Illegal axis identifier

SAI (Set Axis Identifier)

Description: Sets the axis identifiers

Format: SAI *oldaxisid*[ ]*newaxisid*[[ ]*oldaxisid*[ ]*newaxisid*]

Parameters: *oldaxisid*: axis identifier to be renamed, *newaxisid*: new axis identifier, e.g. SAI AXBYCZ: the axes ABC are renamed to XYZ

Response: none

Troubleshooting: Illegal axis identifier or duplicate axis identifier

SAI? (get axis identifier)

Description: Gets the axis identifiers

Format: SAI?

Parameters: none

Response: Axis identifiers of all connected axes

Example:

Send: SAI?

Report: DCBA

SAV (SAVe axis parameters)

Description: Saves the axis parameters in RAM (as modified by the SPA command)

Format: SAV[A][B][C]

Parameters: A,B,C,D: axis identifiers

Response: none

Troubleshooting: Illegal axis identifier

SCA (Set Cursor controlled Axes)

Description: Specifies which two axes are to be controlled by the cursor control keys on the controller keyboard.

Format: SCA *axisid1*[SP]*axisid2*

Parameters: *axisid1*: axis to be controlled by the → and ← keys.
axisid2: axis to be controlled by the ↑ and ↓ keys.
The step width for each keystroke is set by the SST command

Response: none

Troubleshooting: Illegal axis identifier

SCA? (get cursor controlled axes)

Description: Gets the two axes which are controlled by the cursor control keys on the controller keyboard.

Format: SCA?

Parameters: none

Response: two axis identifiers,
the first is controlled by the → and ← keys.
the second is controlled by the ↑ and ↓ keys.

SJA (Set Joystick controlled Axes)

Description: Specifies which two motor axes are to be controlled by the two axes of a directly connected joystick (older models only)..

Format: SJA Joystick_axis [ ]motor_axis

Parameters: Joystick _axis: A: horizontal, B: vertical.
motor_axis: axis identifier of the motor axis to be controlled by the preceeding joystick axis.

Response: none
Example:
SJA ACBD : horizontal motion of the joystick controls axis C, vertical motion of the joystick controls axis D

Troubleshooting: Illegal axis identifier

Remark: Only older models have a game port for direct joystick connection. See the *PIMikroMove* manual for use of joystick on the host PC.

SJA? (get joystick controlled axes)

Description: Gets the two axes which are controlled by a joystick connected directly to the controller (older models only). The motor axis controlled by vertical joystick motion is listed first.

Format: SJA?

Parameters: none

Response: two axis identifiers.

SMO (Set Motor Output)

Description: Sets the motor output directly (servo-control must be set to OFF, e.g. with the SVO command)

Format: SMO A*vvvv*[B*vvvv*][C*vvvv*]

Parameters: A,B,C,D axis identifiers, *vvvv*: value, range: -32767 to 32767

Response: none
Example: SMO A1000
Caution: in servo-off mode the limit switches do not stop the motion!

Troubleshooting: Illegal axis identifier, parameter out of range, servo ON

SMO? (Get Motor Output)

Description: Gets the value in the motor output register. In servo-on mode the actual value, set by the controller, is reported. In servo-off mode the value set by the SMO command is reported.

Format: SMO? [A][B][C]

Parameters: A,B,C,D axis identifiers

Response: Value in the motor command register

Troubleshooting: Illegal axis identifier

SPA (Set Parameter)

Description: Sets specified parameters of the specified axes.

Format: SPA A*index x.x*[B*index x.x*][C*index x.x*][D*index x.x*]

Parameters: A,B,C,D: axis identifier

index: the index of the parameter (for the PID controller); the following values are allowed:

1 for the P-Term (0 to 32767)
2 for the I-Term (0 to 32767)
3 for the D-Term (0 to 32767)
4 for the I-Limit (0 to 32767)
5 for the Velocity feedforward (0 to 32767)
7 for the motor bias (-32768 to 32767)
8 for the maximum position error (0 to 32767)
9 for the maximum value for the motor output (0 to 32767)
10 for the maximum allowed velocity (- 1.79769313486231E308 to 1.79769313486231E308)
11 for the maximum allowed acceleration (- 1.79769313486231E308 to 1.79769313486231E308)
13 for the maximum allowed Jerk (- 1.79769313486231E308 to 1.79769313486231E308)
14 for the numerator of the counts per physical unit factor (1 to 2147483647)(factor = num./denom.)
15 for the denominator of the counts per physical unit factor (1 to 2147483647)(factor = num./denom.)

x.x: the parameter value. Be sure to put one space between *index* and x.x

Response: none

Troubleshooting: Illegal axis identifier

Example: SPA A1 100 A2 25 A3 200 B1 150 B2 35 B3 300

Sets P=100 for Axis A
I=25 for Axis A
D=200 for Axis A
P=150 for Axis B
I=35 for Axis B
D=300 for Axis B

SPA? (Get Parameter)

Description: Gets PID parameters of the specified axes.

Format: SPA? *Aindex* [B*index*][*Cindex*][D*index*]

Parameters: A,B,C, D: axis identifier,

index: the index of the parameter, see list above in SPA command for allowable index values

Response: Requested parameter(s)

Troubleshooting: Illegal axis identifier

Example: SPA? A1 A2 A3

Response:
A1=100
A2=25
A3=200

SRA (Set RAtio)

Description: Sets the electronic gear ratio

Format: SRA A*v.vv*[C*v.vv*]...

Parameters: A,C: axis identifier, *v.vv* value to set

Response: none

Troubleshooting: Illegal axis identifier

SRA? (get ratio)

Description: Gets the electronic gear ratio

Format: SRA?[A][B][C][D]

Parameters: A,B,C,D: axis identifier

Response: the electronic gear ratio values of the requested axis

Troubleshooting: Illegal axis identifier

SSL (Set Soft limit on or off)

Description: Set soft limits (set by the commands NLM and PLM) on or off for the specified axes

Format: SSL A[ ]on | off [B[ ]on | off[□ C[ ]on | off]...

Parameters: A,B,C,D: axis identifier, on | off: keywords for on or off

Response: none

Troubleshooting: Illegal axis identifier

Example SSL A on B off
“1” can be used instead of “on”, “0” instead of “off” ”
SSL A1 is equivalent to SSL A on

SSL? (Get Axes with Soft Limit ON)

Description:	Get axes with soft limit ON
Format:	SSL?
Parameters:	none
Response:	Identifiers of all axes with soft limits “on”

SSN? (Get serial Number)

Description:	Get serial number
Format:	SSN?
Parameters:	none
Response:	serial number

SST (Set Step Size)

Description:	Set step size for cursor control or joystick position control. The values are interpreted as floating point format. Axes not specified retain their previous step-size setting.
Format:	SST A *v.vv*[B*v.vv*][C*v.vv*]...
Parameters:	A,B,C,D axis identifiers, *v.vv* value [physical units]
Response:	none
Troubleshooting:	Illegal axis identifiers

SST? (Get Step size)

Description:	Get step size
Format:	SST?[A][B][C][D]
Parameters:	A,B,C,D axis identifiers
Response:	step size
Troubleshooting:	Illegal axis identifiers

STA? (get STAtus)

Description:	
Format:	STA? [A][B][C][D]
Parameters:	A,B,C, D: axis identifier
Response:	Status Word
Troubleshooting:	Illegal axis identifiers
Remarks:	The status word for each axis is a 16-bit register containing the following information (bit encoding is 0 = LSB, 15 = MSB):

Bit # Description

0 Motion complete flag. This bit is set (1) when the axis trajectory has completed. This flag is only valid for the S-curve, trapezoidal, and velocity contouring profile modes.

1 Wrap-around condition flag. This bit is set (1) when the axis has reached one end of its travel range and has wrapped to the other end of the travel range. Specifically, when traveling in a positive direction past the position +1,073,741,823, the axis will wrap to position -1,073,741,824, and vice-versa. The bit can be reset with the CLR command.

2 Breakpoint reached flag. This bit is set (1) when one of the breakpoint conditions has occurred.

3 Index pulse received flag. This bit is set (1) when an index pulse has been received.

4 Motion error flag. This bit is set (1) when the maximum position error is exceeded. This bit can only be reset when the axis is no longer in a motion error condition

5 Positive limit switch flag. This bit is set (1) when the positive limit switch goes active.

6 Negative limit switch flag. This bit is set (1) when the negative limit switch goes active.

7 Command error flag. This bit is set (1) when an erroneous command has been received by the motion control chip.

8* Servo-control on/off status (1 indicates on, 0 indicates off).

9* Axis on/off status (1 indicates on, 0 indicates off). The C-848 always has the axis ON.

10* In-motion flag. This bit is continuously updated and indicates whether or not the axis is in motion: 1 indicates axis is in motion, 0 not in motion.

11* Reserved (may contain 0 or 1)

12*, 13* Current axis # (13 bit = high bit, 12 bit = low bit). Axis encoding is as follows:

Bit 13	Bit12	MC Axis	C-848 Axis
0	0	1	A
0	1	2	B
1	0	3	C
1	1	4	D

14, 15 Reserved (may contain 0 or 1)

STE (STEp function)

Description:	Performs a step and records up to 1024 positions as the axis responds
Format:	STE A*v.vv* [*cycles*]...
Parameters:	A: axis identifier, cycles: delay (in servo-loops) between the recorded positions
Response:	none
Troubleshooting:	Illegal axis identifiers

NOTE

The current version of *PIMikroMove* does not support the Step Response dialog with this controller. Use C-*848Control* for convenient display of the results of this command.

STE?

Description:	Lists the recorded positions
Format:	STE? As *n*...
Parameters:	A:axis identifier, s: Startindex (0-1023) n: number of values to be read (1-1024-s)
Response:	A=v.vv A=v.vv
Troubleshooting:	Illegal axis identifier, no step response values stored, too many values requested

STP (Stop Motion)

Description:	Stops the current motion of all axes; same as HLT addressed to all axes. Does not work during MPL, MNL or REF motion (use #24 instead). After the STP command the system does not require re-initialization. Any target position can be subsequently reached by the MOV or MVR command, and the POS? command gives the actual position. STP sets the ERR code to 10.
Format:	STP
Parameters:	none
Troubleshooting:	Communication breakdown

SVO (set SerVO on or off)

Description:	Sets servo-control mode on or off (closed-loop/open-loop mode).
Format:	SVO A*n*[B*n*][C*n*]
Parameters:	A,B,C,D axis identifiers, *n*=1 (or “on”) : Set servo on *n*=0 (or “off”): Set servo off
Response:	none
Troubleshooting:	Illegal axis identifier

SVO? (get servo status)

Description:	Reports the servo mode of the specified axes
Format:	SVO?[A][B][C]
Parameters:	A, B, C: axis
Response:	1: Servo on 0: Servo off

Example	SVO? ABC
	A=1
	B=1
	C=1

Troubleshooting:	Illegal axis identifier

TIM? (Tell System Time)

Description:	Reports the number of servo-loop counts since controller start-up on the first motion-control board, if available
Format:	TIM?
Parameters:	none
Response	system time (in servo-loop counts; the duration of one loop is 400 µs)

TIO? (Tell Digital I/O Lines)

Description:	Tell number of installed digital I/O lines
Format:	TIO?
Parameters:	none
Response:	I=N (N=Number of installed input lines) O=N (N=Number of installed output lines)

TMN? (Tell Minimum Travel Value)

Description:	Tell minimum position value in physical units
Format:	TMN?[A][B][C][D]
Parameters:	A,B,C,D axis identifiers
Response	Value of the low end of the travel range (either the position of the negative limit switch, or the value of the negative soft limit, if set)

TMX? (Tell Maximum Travel Value)

Description:	Tell maximum position value in physical units
Format:	TMX?[A][B][C][D]
Parameters:	A,B,C,D axis identifiers
Response	Value of the high end of the travel range (either the position of the positive limit switch, or the value of the positive soft-limit, if set)

TNJ? (Tell Number of joysticks)

Description:	Tell number of joysticks connected directly to the controller
Format:	TNJ?
Parameters:	none
Response	0 or 1

TVI? (Tell Valid axis Identifiers)

Description:	Tell valid axis identifiers
Format:	TVI?
Parameters:	none
Response:	String with allowed axis identifiers
Troubleshooting:	

VEL (Set Velocity)

Description:	Set velocity in physical units / sec
Format:	VEL A *v.vv*[B*v.vv*][C*v.vv*]
Parameters:	A,B,C,D axis identifiers *v.vv*: velocity value
Response:	none
Troubleshooting:	Illegal axis identifiers

VEL? (Get Velocity)

Description: Reports the programmed velocity (rounded to 3 digits).
[physical units/sec]

Format: VEL?[A][B][C]

Parameters: A,B,C,D axis identifiers

Response: velocity

VER? (Get Version)

Description: The systems reports the firmware version of the controller.

Format: VER?

Parameters: none

Response: Firmware version e.g.: 1.10 1704030930

VMO (Virtual Motion)

Description: Virtual Motion

The system answers whether or not the target position is out of reachable. The command can be used to check the working space

No motion occurs

The current target position does not change.

Format: VMO A*v.vv*[B*v.vv*][C*v.vv*]...

Parameters A,B,C,D: axis identifiers
v.vv Absolute position value [physical units]

Response: 0 for target position can be reached

1 for target position is out of reach

VST? (Get available Stages)

Description: The systems reports the names of stages selectable by the CST command

Format: VST?

Parameters: none

Response: List of available stages, each on a new line

WAA (Wait for Completion of all axes)

Description: Checks wether all axes have completed their current motion within the specified time.

Format: WAA value

Parameters: *value*: time in milliseconds

Response: 0 : motion not completed
1 : motion completed
Example : WAA 1000
If all axes are on target within the next second a 1 is reported otherwise 0

WAI (Wait for Completion)

Description: Hold next command (single-character commands excepted) until all specified axes have completed their current motion.

Format: WAI [A][B][C]

Parameters: *axis*: axis identifier

Response: none

Troubleshooting: Illegal axis identifier

#5 (Poll Motion Status)

Description: Reports the current motion status.

Format: #5 (single ASCII character number 5)

Parameters: none

Response: Hexadecimal sum of the following codes for all axes which are moving:
1 axis A (is moving)
2 axis B (is moving)
4 axis C (is moving)
etc.

Examples: 0 indicates motion of all stages complete
7 indicates axes A,B and C are moving

#6 (Position Change?)

Description:	Asks if position has changed since last POS? query.
Format:	#6 (single ASCII character number 6)
Parameters:	none
Response:	Hexadecimal Sum of the following codes for all axes whose position has changed: 1 axis A (has changed) 2 axis B (has changed) 4 axis C (has changed) etc. Example: A indicates that axes B and D have moved

#7 (Controller Ready?)

Description:	Test if Controller is ready to perform a new command
Format:	#7 (single ASCII character number 7)
Parameters:	none
Response:	B1h (ASCII character 177) Controller is ready B0h (ASCII character 176) Controller is not ready (e.g. performing a FSC or REF command)

#8 (Macro running?)

Description:	Test if a macro is running
Format:	#8 (single ASCII character number #8)
Parameters:	none
Response:	0 (ASCII character 48) no macro is running 1 (ASCII character 49) a macro is running

#24 (Stop all axes)

Description:	Stop All Axes; single-character alias for STP except that it also works during MPL, MNL and REF motion.
Format:	#24 (single ASCII character number 24)
Parameters:	none
Response:	none

#27 (ESC) (System Abort)

Description:	ASCII Character #27 aborts all system activity. It can be used as an EMERGENCY FULL-SYSTEM STOP. Motion of all axes stops immediately, all servo-registers are reset, the servo-loop is turned off. The controller emits a continuous beep at 400 Hz. For restart, the controller must be reset manually or turned off and on (power cycled).
Format:	#27 (single ASCII character number 27)
Parameters:	none
Response:	none

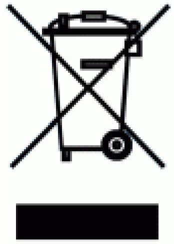
10 Old Equipment Disposal

In accordance with EU directive 2002 / 96 / EC (WEEE), as of 13 August 2005, electrical and electronic equipment may not be disposed of in the member states of the EU mixed with other wastes.

To meet the manufacturer’s product responsibility with regard to this product, Physik Instrumente (PI) GmbH & Co. KG will ensure environmentally correct disposal of old PI equipment that was first put into circulation after 13 August 2005, free of charge.

If you have such equipment from PI, you can send it to the following address postage-free:

Physik Instrumente (PI) GmbH & Co. KG
Auf der Römerstr. 1
76228 Karlsruhe, Germany



11 Technical Data

Model	**C-848.23 / C-848.23i**	**C-848.43 / C-848.43i**
Function	DC motor controller	DC motor controller
Drive type	DC motors Voice coil linear drives	DC motors Voice coil linear drives
Channels	2	4
Motion and control		
Servo characteristics	Programmable 32-bit PID servo-algorithm, V-ff filter, 100 µs/active axis; on-the-fly parameter changes	Programmable 32-bit PID servo-algorithm, V-ff filter, 100 µs/active axis; on-the-fly parameter changes
Trajectory profile modes	Linear interpolation, trapozoidal, S-curve, electronic gearing	Linear interpolation, trapozoidal, S-curve, electronic gearing
Microstep resolution		
Processor	Dual processor CPU 133 MHz Motion chip, 2.5 kHz servo- update rate	Dual processor CPU 133 MHz Motion chip, 2.5 kHz servo- update rate
Encoder input	A/B, TTL-level, differential, 5 MHz	A/B, TTL-level, differential, 5 MHz
Stall detection	Motor-stop, or when programmable position error is exceeded	Motor-stop, or when programmable position error is exceeded
Limit switches	2 per axis, TTL, also soft limits programmable by axis	2 per axis, TTL, also soft limits programmable by axis
Origin switches	1 per axis, for real-time reference point	1 per axis, for real-time reference point
Motor brake	TTL-level, programmable	TTL-level, programmable
Control In		
Electrical properties		
Operating voltage	100 to 240 VAC, 50 to 60 Hz	100 to 240 VAC, 50 to 60 Hz
Output power per channel	Analog H-bridge; 0 to ±12 V, 5 W/channel, 12-bit DAC, 10-bit, 24.5 kHz output for PWM amp	Analog H-bridge; 0 to ±12 V, 5 W/channel, 12-bit DAC, 10-bit, 24.5 kHz output for PWM amp
Output voltage per channel	0 to ±10.5 V analog TTL-level for PWM-mode SIGN and MAGN signals	0 to ±10.5 V analog TTL-level for PWM-mode SIGN and MAGN signals
Current per channel		
Current limitation	1 A max. (short-circuit proof)	1 A max. (short-circuit proof)
Interfaces and operation		
Communication interfaces	RS-232 standard, cable included; IEEE 488.2 (GPIB) (C-848.42i only)	RS-232 standard, cable included; IEEE 488.2 (GPIB) (C-848.43i only)
Motor connector	Sub-D connector 15-pin	Sub-D connector 15-pin
Controller network	via IEEE 488 (GPIB) option	via IEEE 488 (GPIB) option
I/O ports	8 TTL inputs, 8 TTL outputs	8 TTL inputs, 8 TTL outputs
Command set	PI GCS	PI GCS
User software	*C-848Control* and *PIMikroMove*™ operating software	*C-848Control* and *PIMikroMove*™ operating software
Software drivers	LabView™ drivers, libraries for C, Pascal and BASIC under Windows	LabView™ drivers, Libraries for C, Pascal and BASIC under Windows
Supported functionality	Autostart macro, macro programming Monitor and keyboard connections Motor brake control output	Autostart macro, macro programming Monitor and keyboard connections Motor brake control output
Manual control	Joystick via host PC,	Joystick via host PC
Miscellaneous		
Operating temperature range	+10 °C to +50 °C	+10 °C to +50 °C
Mass	8.4 kg	8.2 kg
Dimensions	447 x 450 x 90 mm (19-inch rack-mountable or desktop)	447 x 450 x 90 mm (19-inch rack-mountable or desktop)

12 Front and Rear Panel Elements

12.1 Front Panel

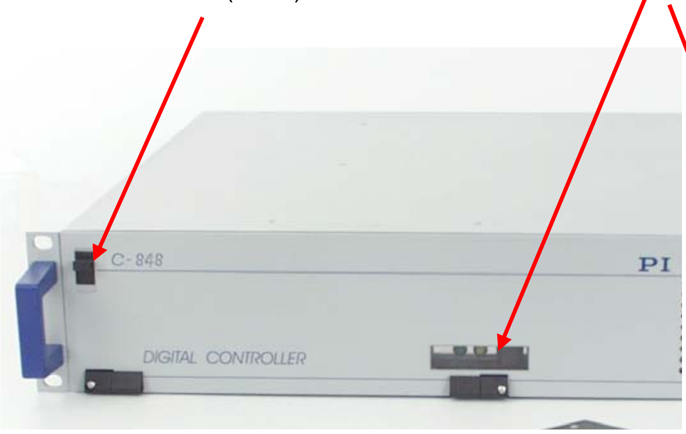


Fig. 3. C-848 *front panel with LED window at bottom center.*

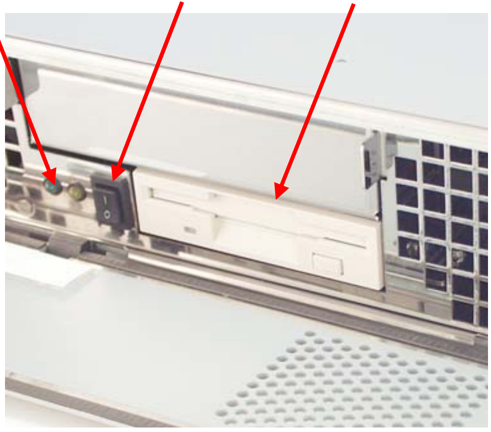


Fig. 4. Front panel opened, exposing diskette drive and 1 of the 2 power switches

Only the amber and green LEDs are visible with the front panel closed. They indicate power-on and error status respectively.

Sliding the two latches at the top right and left corners of the front panel allow it to opened as shown, exposing the diskette drive and the front-panel ON/OFF switch (it and the rear-panel OFF/ON switch are wired in series).

12.2 Rear Panel

All connections are on the rear panel. They include the power connection, the connection to the host computer (RS-232, RS-422 and, if present, joystick and/or an IEEE 488 GPIB port), digital I/O for external signals and a connector for each controlled motor. There are also connectors for a keyboard and VGA monitor. Keyboard and monitor are not normally required, but may be useful if problems occur. Pinouts are industry standard.

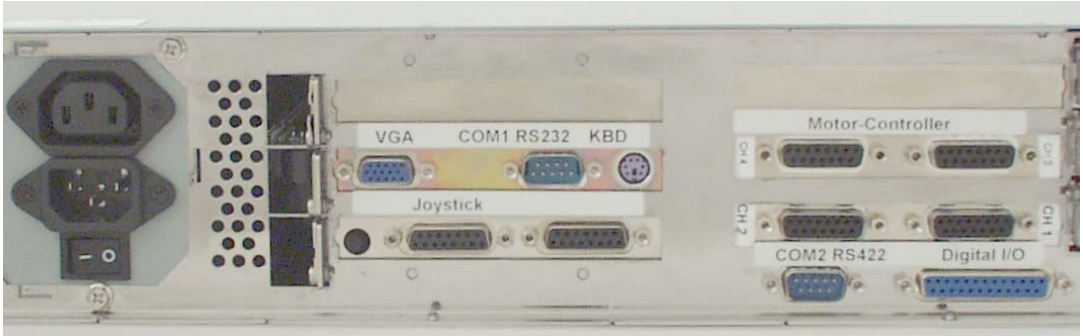


Fig. 5 C-848 *rear panel; current and future versions lack the joystick connection*

NOTE

The rear-panel OFF/ON switch is wired in series with the switch behind the front panel. Both switches must be ON for the unit to operate

12.2.1 Motor Connectors

Each motor axis connects with a sub-D 15-pin connector having the following pinout:

1	Brake, active low, TTL level (5 V = brake off)
9	Motor (-)
2	Motor (+)
10	Power GND
3	MAGN (PWM-TTL)
11	SIGN (TTL)
4	Output +5 V
12	Negative limit (active high)
5	Positive limit (active high)
13	REFS
6	Limit GND
14	Encoder: A(+) / ENCA
7	Encoder: A(-)
15	Encoder: B (+) / ENCB
8	Encoder: B (-)

12.2.2 Digital I/O Connector

Eight input and eight output lines are available on a sub-D 25f connector. Do not confuse the digital I/O channels (A-H) the axes (A-D), they are completely independent.

1		GND
	14	Input A
2		Input B
	15	Input C
3		GND
	16	Input D
4		Input E
	17	Input F
5		GND
	18	Input G
6		Input H
	19	DINCLK
7		GND
	20	Output A
8		Output B
	21	Output C
9		GND
	22	Output D
10		Output E
	23	Output F
11		GND
	24	Output G
12		Output H
	25	DOUTCLK
13		GND

Latch Mode

The input register can be latched by external signals at DINCLK. Pulling down this line externally, the input levels are latched and held. A new valid data byte can be latched by positive strobes.

Transparent Mode

In the transparent mode the DINCLK line is not serviced, any signals at the input lines can be read as long as they are there. No latching occurs.

Max. current ratings:
Sink: 24 mA,
Source: 2.6 mA

12.2.3 Optional Keyboard and Monitor

Keyboard: standard keyboard with PS/2-style (mini-DIN) connector. Firmware may be configured for US keyboard layout.

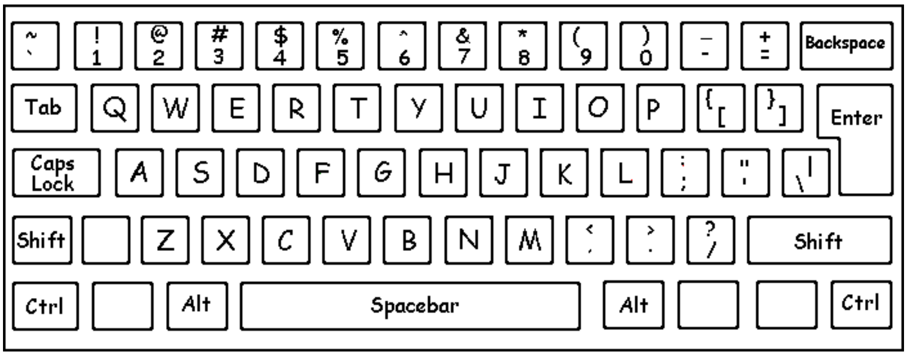

~ ` ! 1 @ 2 # 3 $ 4 % 5 ^ 6 & 7 * 8 (9) 0 _ - + = Backspace
Tab Q W E R T Y U I O P { [}] Enter
Caps Lock A S D F G H J K L : ; " ' | \
Shift Z X C V B N M < , > . ? / Shift
Ctrl Alt Spacebar Alt Ctrl

Fig. 6 US keyboard layout

Monitor: standard VGA monitor with sub-D15 three-row connector