U s e r   M a n u a l

MS 52E User Manual

C-844

Precision DC-Motor Controller

R *5.00*

Operating Instructions
and
Technical Documentation

This Document is valid for the Products:

C-844.20	Precision DC-Motor Controller
C-844.40	Precision DC-Motor Controller

Firmware 5.xx

Release: *5.00*
Release Date: *2002-02-18*

Table of Contents :

This operating manual complies with firmware version 5.0 and higher.

© Copyright 2002 by Physik Instrumente (PI) GmbH

Release: 5.00
File:MS52E500.doc, 741376 Bytes
MDate: Thursday, 28. February 2002, 10:49

0 Manufacturer Declarations

0.1 Declaration of Conformity

The manufacturer,

Physik Instrumente (PI) GmbH
auf der Roemerstrasse 1
D-76228 Karlsruhe

declares, that the product ***C-844 Precision DC-Motor Controller*** complies with these specifications:

EMC:	EN55022 (1991), Group1, Class B	
	EN50082-1 (1992) / IEC 801-2:1991	(4 kV Contact Discharge) (8 kV Air Discharge)
	EN50082-1 (1992) / IEC 801-3: 1984 (3V/m)	
	EN50082-1 (1992) / IEC 801-4: 1988 (1 kV power lines, 0.5 kV Signal lines)	
Safety:	IEC 1010-1:1990+A1 / EN61010-1:1993 (Low voltage Directive)	

The product complies with the requirements of the EMC Directive 89/336/EEC and CE markings have been affixed on the devices.

0.2 Quality and Warranty Clauses

Certification
PI (Physik Instrumente) certifies that this product met its published specifications at the time of shipment. The device was tested with DC-Motors and stages specified in the PI product catalogs.

Warranty
This Physik Instrumente product is warranted against defects in materials and workmanship for a period of one year from date of shipment. Duration and conditions of warranty for this product may be superseded when the product is integrated into (becomes a part of) other Physik Instrumente products. During the warranty period, Physik Instrumente will, at its option, either repair **or** replace products which prove to be defective.

Limitation of Warranty
The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by the buyer, buyer supplied products or interfacing, unauthorised modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

The design and implementation of any circuit in this product is the sole responsibility of the buyer. **PI** does not warrant the buyer's circuitry or product malfunction that result from the buyer's circuitry. In addition, **PI** does not warrant any damage that occurs as a result of the buyer's circuit or any defects that result from buyer-supplied products.

No other warranty is expressed or implied. Physik Instrumente specifically disclaims the implied warranties of merchantability and fitness for a particular purpose.

1 Introduction

C-844 Precision Motor Controller is a 4-channel desk top DC-Motor Controller for motors up to 6 Watt power consumption. Primary applications of the C-844 are found in industrial production and test facilities as well as in laboratory experimental setups. Every place where precise positioning combined with demanding performance of servo controlled motion is required, the C-844 delivers an adequate solution.

The C-844 is based on a multi-processor architecture including a fast DSP motion control chipset for digital position servo control and a host processor for communication and command execution. Additional features like SCPI command parsing, IEEE 488.2 interfacing and macro command execution are also performed by the host processor. An internal real time operating system based on a multitasking architecture handles interrupts, I/O operations, limit switches and user requests.

The C-844 offers high performance PID motion control with many options of trajectory generation and filter setting. The motors can be controlled in position, velocity, acceleration and other motion relevant parameters.

For position feedback, incremental rotary or linear encoders can be used. Standard PI motors perform a resolution of 2000 counts/rev and are well suited for high resolution positionings. Even better absolute position accuracy can be achieved by linear incremental scales, available as option for the M-500 series positioning stages.

C-844 Features :

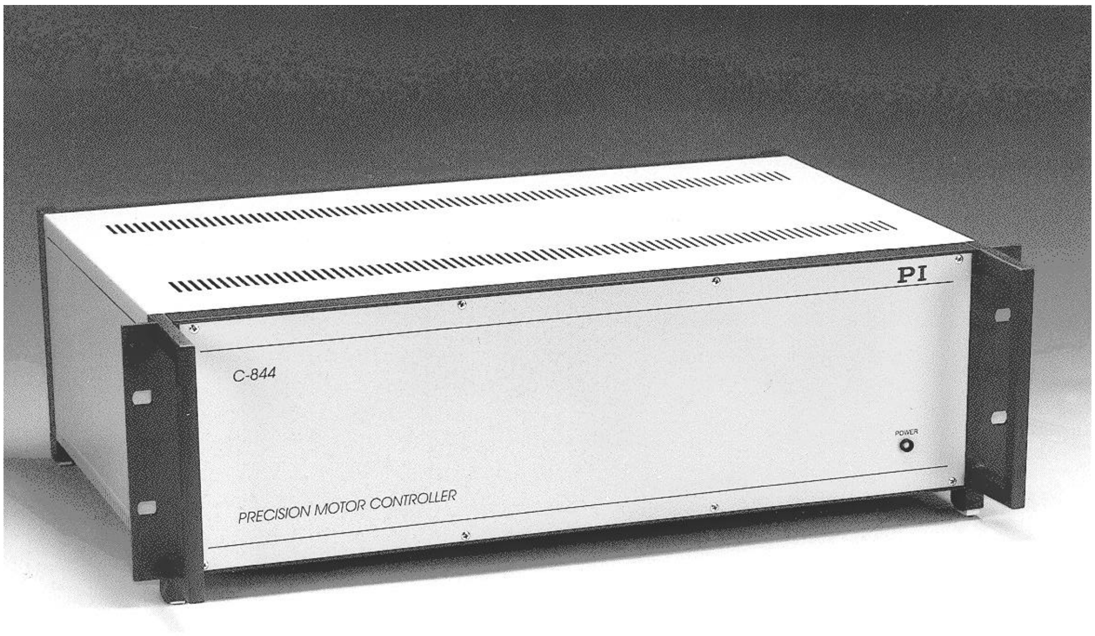
- 4-Axes DC-Motor Controller (19" Desk top device)
- Motor power up to 6 Watts (average)
- Quadrature encoder input
- 12 bit analog or PWM output
- P-I-D-Vff motion control
- Profile modes: trapezoidal, S-curve, velocity
- Electronic gearing
- Limit Switch control
- 16 TTL I/O channels
- IEEE-488.2 and RS-232 interfaces
- SCPI command language
- Macro command capability

Applications

The C-844 is a versatile motion and positioning controller for motorized stages DC-Mike drives and other motorized devices. With its additional I/O features like analog and digital inputs and trigger signal handling, the C-844 offers all properties required for industrial use.

Typical Application are:

- **Linear and rotary micropositioning stage motion control**
- **General laboratory automation**
- **Production robotics**
- **controlling automatic alignment test stands**
- **Quality inspection automation**
- **Object handling and alignment**
- **Field scanning**
- **Path tracking**
- **Laser cutting**



C-844 DC-Motor Controller

1.1 C-844 Specifications

Model:	C-844.00
Function:	4-Channel DC-Motor Controller
Motor Types:	C-120.80, C-124.50, C-128.50
Max. motor power (per channel):	6 W
Output Amplifiers:	Analog H-bridge, 12 bit, (optional PWM output, 10 bit, 24.5 kHz for external amplifiers)
Output Voltage:	12 V
Current Limitation:	2.5 A
Output Safety Options:	Amplifier switchable by command, automatic overload shutdown function
Internal Safety Options:	Temperature and voltage control
Heat sink:	Internal heat channel with fan
Encoder (for position feedback):	Quadrature encoder signals, optional single ended TTL or RS-422 (differential mode), software configurable.
Max. encoder counting rate:	10^6 count/s
I/O ports:	8 TTL inputs, 8 TTL outputs
Analog Input:	4 channels, 5V unipolar, 10 bit ADC
Limit Switches:	2 limit switches per axis(one for each direction of travel, software programmable polarity, pull-up/pull-down
Reference signal detection:	4 reference inputs, encoder synchronized
Motor brake output:	15 V, software controllable
Trajectory monitor output:	Frequency/Voltage conversion
CPUs:	16 bit CPU H8/10MHz DSP MC1401A
Sample Rate:	400 µs
Position range:	-1,073,741,824 to 1,073,741,823 counts
Filter mode:	P-I-D+Vff+bias
Communication Interfaces:	RS-232 (2x), IEEE 488.2
Command Set:	SCPI command set, ASCII format
Trajectory control.	P-I-D-Vff motion control
Profile modes:	trapezoidal, S-curve, velocity contouring, electronic gearing
Standard options:	- Macro command capability
Operating Voltage:	90 to 240 Vac, 50 to 60 Hz
Fuse:	0.8 A, slow
Chassis:	19" rack mountable
Dimensions:	Height: 3 HE, Width 19", Depth: 290 mm
Weight:	5.2 Kg

2 Preparing Operation

2.1.1 Unpacking and Inspection

The C-844 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-844 order:

- C-844 DC-Motor Controller
- Line cord
- RS-232 Null Modem Cable (C-815.34)
- PI Motion CD
- MS 52E User Manual (this document)

2.1.2 Power Requirements

The C-844 has a wide range input power supply that can be used from **100 VAC to 240 VAC at 50 to 60 Hz**. Connect the line cord to the AC receptacle on the rear panel of the device.

2.1.3 Power-On Defaults

During power-on reset the C-844 is initialized to a defined default state. Although the default settings are not matched to specific motor configuration, the motor can be moved instantly by a move command. For better adapted motion performance, individual parameter values have to be applied.

The same default state can be invoked by sending the "*RST" command. All 4 axes are set to the same default state.

Default settings:

1. Position counters are set to 0
2. Trapeze profile mode activated
3. Servo loop ON
4. Output set to DAC
5. Motor current amplifiers ON
6. Motor brake output OFF
7. Target position set to 0 (servo controlled)
8. Encoder set to differential mode
9. Interface state set to "no interface selected". LED at front panel shines yellow. The first character received determines the interface and the interface state is set to either "RS232" or "IEEE". The LED turns green.
10. All limit switch lines are terminated by pull-up resistors.
11. Limit switches are initialized by the autodetect function. None limit switch should be tripped during power on.
12. P-I-D filter parameters are set to 400-10-1000
13. Software moving range limits are set OFF
14. Max. velocity is set to 6000 c/s (trapeze and S-curve modes)
15. Max. acceleration set to 20 (==50000 c/s^2), (for trapeze, S-curve and velocity modes)
16. Jerk: 0.001 (== 15.625.000 c/s^2/s)

2.2 Communication

2.2.1 RS-232 Interface

The serial communication port is accessed via the "RS-232" connector at the rear panel. The port is set to these parameters:

9600 baud, 8 bits, no parity,   Handshake: RTS/CTS

2.2.2 GPIB-488 Interface

The C-844 complies with IEEE488.2 standards and supports status register structure as described in chapter "C-844 Register Structure".

Function implementation list :

SH1	Full Source Handshake capability
AH1	Full Acceptor Handshake capability
T6	Primary Address, Serial Poll, Talker de-address at MLA
L4	Primary Address, Listener de-address at MTA
SR1	Full Service Request capability
PP0	no parallel polling
DC1	Full Device Clear capability
DT0	no trigger functions
C0	no controller functions

Input and output buffer have each 128 bytes . The maximum length of a command line send to the C-844 must not exceed the 128 bytes. Also the maximum length of a response does not exceed 128 bytes.

Multiple commands can be assembled within on command line, each single command separated by a ";".

If one or more queries are sent in one command line, then each query requires one individual reading access. The interface will be unable to accept new commands before the last response is retrieved.

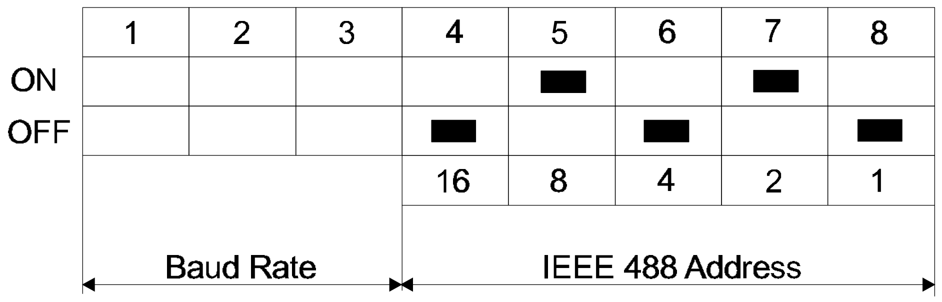
The MAV bit in the status byte indicates that data are in the output queue.

Termination of Communication

Communication is terminated by the EOS character "new line" decimal 10 and the EOI signal line.

Dip Switch Setting:

The 8-bit DIP switch at the rear panel is used for IEEE 488 address setting and optional baud rate selection.



Default Address: 10

2.2.3 GPIB Programming Notes

With firmware version 5.00 the IEEE-488 communication routines inside the C-844 firmware have been redesigned and adapted to comply with fast PC communication.

A stable communication with the GPIB interface is based on some basic rules that has to be considered when sending and receiving data strings.

The C-844 offers the feature to process compound command lines, consisting of multiple single commands, separated by a semicolon. The *WAI command can be used to sequence consecutive moves. It waits until the system has terminated a move, or in more general, until all pending operations are completed. This feature may conflict with the rule of synchronous communication, that means the device may send a report anytime, depending on the duration of a move. The following command

"TARG:RPOS 120000; *WAI; AXIS:POS?"

causes the C-844 to reports the axis position after the move of 120000 counts is completed. This would probably create a timeout error at the PC side. Also the C-844 will not be able to accept further commands as long as the report string is not taken.

These conflicts can be solved by reading the MAV bit from the status byte (ibrsp, *STB?) before writing and reading operations in order to determine the availability of data. The reading routine (ibrd) must only be called if there is something in the C-844 output buffer or if the programmer is sure that within the timeout period there will be something to read.

3 Quick Start

1. Connect the stages/motors with the C-844.
   NOTE: Every time you change the stage wiring, shut the C-844 down. System operation will fail if stages are disconnected during operation.
2. Connect the communication cable, either RS232 or GPIB. Use a null-modem cable and connect it with connector "RS-232" for serial communication or connect the standard GPIB bus.
3. In case of the GPIB communication, select the device address using the DIP switch address switch at the rear panel.
4. Install the software, either WinTerm or DCMove or any other communication program.
5. Turn the power on. After power on, all parameters are set to the power on default values. The LED on the front panel lights yellow, indicating that a power on reset was performed and no interface is yet selected.
6. Start the program
7. Type the command "*IDN?" asking for device identification message.
   The response looks like : "**`Physik Instrumente, C-844, 277,5.1/2.10`**"
   Serial number: 277, firmware release:5.1, processor version: 2.10
8. The first character received determines which interface is to be used. The other port is disabled until the next power-on reset or reset by "*RST" command. The front panel LED changes to green, indicating that an interface is selected.
9. Check whether the limit switches are initialized correctly. If during power-on one limit switch was tripped (error condition), the limit switch state of all axes are set to OFF. This state can be questioned with "AXIS:LIMIT?". If the response is "OFF", at least one axis might be standing on one limit switch. In this case, switch the controller off and move the axes by hand out of the limit switch and turn the controller on again.
10. Start a move: type "AXIS 1; TARG 5000", Motor #1 moves to position 5000.

3.1 Limit Switches

The C-844 support motion travel limit switches that can be used to automatically stop the motor and to move the stage backwards until the limit switch is released.

Any time a limit switch is tripped and the sense line changes its voltage level, a limit handling routine is started and drives the motor backwards until the limit switch is released.

The limit switch sensing mechanism can be set to ON or OFF by command:

`"AXIS:LIM:STAT ON"`	Activates limit switch sensing mechanism for all 4 axes. **(default after Power-on)**
`"AXIS:LIM:STAT OFF"`	Disables limit switch sensing mechanism for all 4 axes.

Each motor axis has 2 limit switch lines, one for the positive, the other for the negative limitation of travel. Both lines for each axis can be terminated inside C-844 either by pull ups, pull-downs or they can be set open.

The termination can be defined by command:

`"AXIS:LIM:TERM PUPP"`	terminate with pull ups **(default after Power-on)**
`"AXIS:LIM:TERM PDOW"`	terminate with pull downs
`"AXIS:LIM:TERM Z"`	no termination
`"AXIS:LIM:TERM DEF"`	Pull-up (Standard termination)

The voltage level that corresponds with a tripped limit switch can be defined by command:

`"AXIS:LIM:SENS HIGH"`	Handles high level as tripped
`"AXIS:LIM:SENS LOW"`	Handles low level as tripped
`"AXIS:LIM:SENS AUTO"`	Handles the current limit switch level as not tripped and sets the LS detection level according to the current state. During power-on procedure, this command is executed automatically. If the function reads different levels at both limit switches of the same axis, the complete limit switch sensing mechanism is disabled and an error message is generated. If the command is send during operation, only an error message is issued.

After power-on, the "AXIS:LIM:SENS AUTO" command is performed to define the current limit switch levels as the working ones and any transition into another logical level will cause a limit switch interrupt.

Note: *If during power-on one stage is already within a limit switch area and the switch is tripped, both lines of one axis have different levels and the limit switch sensing mechanism of all axes is disabled.*

This state can be recognized by query the error queue with **"SYST:ERR?"** command or by directly asking the limit switch states**: "AXIS:LIM:STAT?"**.

In order to avoid unexpected disabling of the limit switch detection mechanism, the user should verify the limit state every time after power up.

3.1.1 Limit switch handling routine

After a limit switch event was recognized, the axis is set to trapezoidal mode and a move command towards the maximum count number opposite to the tripped LS is performed. When the LS is released, the profile generator is stopped and the old target is and profile mode is restored. The motor is servo controlled at the point where the LS was released.

3.2 Digital I/O

The C-844 offers 8 digital inputs and 8 digital outputs for triggering and synchronizing external devices.

Digital I/O lines can be accessed via the connector "DIGITAL I/O" at the rear panel. For pin assignment see chapter "Appendix/Pin Assignment".

3.2.1 Input Lines

Reading external TTL signals:

PORT:DINput? or PORT:DIN? delivers one byte of information representing the 8 input lines. Each bit set corresponds with a +5V signal at that line. The reported value is a decimal value:

Example: **PORT:DIN?** reports:	“197”
	== hexadecimal: C5
	== binary: 11000101

In this case, a +5V level is applied to channels 1,3,7 and 8.

Hardware note: For digital input an SN74ALS573 is used in conjunction with a 10k pull-up resistor. Additional SN74ALS157 is used with upper 4 bits.

3.2.2 Output Lines

Output TTL Signals :

The 8 TTL output lines are represented by one 8-bit register. Each bit represents one output line. If the bit is set, the corresponding output line is set to +5V.

The command **PORT:DOUT n** writes the decimal value n into the output register. The value written can be verified by reading the value back by the command **PORT:DOUT?**.

Hardware Note: For digital output an SN74ALS273 is used.

3.3 Trigger Commands

Trigger commands can be used to synchronize external devices with motor positions or to start movements and macro commands, to change velocities or electronic gear ratios upon external signals applied to the digital input channels.

8 Trigger commands for channels 1 to 8 can be used to define the action that shall be performed upon receiving a trigger signal:

[PORT:]TRIG:DIN1 *parameter*	Input channel #1, assigned to motor #1
[PORT:]TRIG:DIN2 *parameter*	Input channel #2, assigned to motor #2
[PORT:]TRIG:DIN3 *parameter*	Input channel #3, assigned to motor #3

[PORT:]TRIG:DIN4 ***parameter***	Input channel #4, assigned to motor #4
[PORT:]TRIG:DIN5 ***parameter***	Input channel #5
[PORT:]TRIG:DIN6 ***parameter***	Input channel #6
[PORT:]TRIG:DIN7 ***parameter***	Input channel #7
[PORT:]TRIG:DIN8 ***parameter***	Input channel #8

As parameter one of these keywords can be used:

NONE	Trigger signal is ignored
my_prog	Trigger signal starts macro my_prog
POS	Trigger signal sets new target and starts motor
VEL	Trigger signal sets new velocity
GRAT	Trigger signal sets new gear ratio

Applicable parameters depend on channel:

Command	Possible parameter
TRIG:DIN1 TRIG:DIN2	NONE POS VEL GRAT my_prog
TRIG:DIN3 TRIG:DIN4	NONE POS VEL my_prog
TRIG:DIN5 TRIG:DIN6 TRIG:DIN7 TRIG:DIN8	NONE My_prog

To prepare the trigger function, a target, velocity or gear ratio has to be written to the these axis-related registers. Assuming a move to position 5000 shall be started at the trigger, the target has to be stored in the trigger position register:

[SOUR:]TARG:TRIG:POS 5000

In the same way trigger velocity and trigger gear-ratio values can be defined:

[SOUR:]TARG:TRIG:VEL 85000

[SOUR:]TARG:TRIG:GRAT 3500

As default all these registers contain zero.

3.3.1 Trigger Output

Depending on the in-motion bit of the motion condition register, trigger signals can be output through the digital output channels 1 to 4.

The trigger output is initialized by the command

[PORT:]TRIG:DOUT <value> , with <value> is within 0 to 15.

If the value is 0, the trigger output is disabled.

Example: **PORT:TRIG:DOUT 9**

This command enables trigger output for motor channel #1 and #4. The bits of the motion condition register is applied to the output channels 1 and 4.

When the trigger output mechanism is activated, the end of a motion, defined by the end of the profile trajectory, resets the motion bit in the motion condition register and sets the digital output of that channel to high, which channel number corresponds with the axis number (channel 0 = axis 1.

Any other number as parameter sets the digital output channels according to the binary representation of the number:

[PORT:]TRIG:DOUT 255

This command sets all digital output lines 1 to 4 to the motion condition register and the output channels 5 to 8 to high (+5V).

3.4 Motion Errors

As default, the motion error sensing mechanism is enabled.

If the current position error becomes larger than a programmed maximum error limit, an STOP is applied to that axis.

The motor is servo controlled at the position where the motion error occurred.

The limit switch handling routine sets the profile mode to trapeze, independent which mode has been active before and activates all motion parameters of the now active trapeze mode for that axis.

The motion error limit can be set with that command:

"LIM:PERR <value>"

3.5 Motor Output Limits (torque limitation)

In addition to setting various PID gain values the C-844 also allows the maximum value output by the filter to be set. This motor limit value is set using the command "OUTP:LIM n". It can be read back using the command "OUTP:LIM?".

The specified motor limit affects the filter output such that if the magnitude of the filter output value (positive as well as negative) exceeds the motor limit than the output value is maintained at the motor limit value. Once the filter output value returns below the specified limit than normal servo filter values are output

The motor limit is only applied during closed loop servo operations, when the servo filter controls the motor output value. It does not affect the output motor value applied during open loop operations, set by the OUTPx command.

3.6 Motor Bias

When using an axis which has a net force in one direction or the other (such as a vertical axis which experiences the force of gravity) the motor bias function of the PID compensation filter may be useful. By adding a constant bias value to the filter output, the overall position error of the filter can be reduced by directly compensating for the constant force.

The motor bias value is set using the command **"SOUR:FILT:BIAS n"** It can be read back using the command **"SOUR:FILT:BIAS?"**.

The motor bias value is applied to the filter value at all times the chipset works in closed loop mode (Servo ON). If the chipset transits to open loop mode (**"OUTP:SERV OFF"**), the programmed motor bias value will be output directly to the motor.

Sending manual motor output commands ("OUTP n"), the bias value is not taken into account, so the commanded motor output is independent from the bias setting.

The following example illustrates this: If the C-844 is in closed loop mode with a motor bias value of 100, then if a motor-off command is given ("OUTP:SERV OFF"), then the output motor command will be exactly 100. Thereafter if the host sends a manual motor command of 200 (using the "OUTP n" command), then the output motor command will be 200. At this instant the C-844 is returned to closed loop mode however ("OUTP:SERV ON" command), the motor bias value will again be added to the filter output.

If the specified motor bias value does not properly compensate for the offsetting DC load, then after a ***"OUTP:SERV OFF"*** *command the axis may move suddenly in one direction or another. It is the responsibility of the host to select a motor bias value such that safe motion is maintained.*

3.7 Status Registers

(For status register reference see chapter "C-844 Register Structure")

The C-844 system reflects its operating status by five register sets. Instrument events such as limit switch hits or motion errors are monitored by individual bits in the condition registers and latched in the corresponding event register.

C-844 Register Survey :

Questionable Registers	Questionable Condition Register Questionable Event Register Questionable Enable Register	16 Bit	SCPI - mandated
Operation-Registers	Operation Condition Register Operation Event Register Operation Enable Register	16 Bit	SCPI - mandated
Motion-Registers	Motion Condition Register Motion Event Register Motion Enable Register	16 Bit	C-844-specific
Standard Event Registers	Standard Event Status Register Standard Event Enable Register	8 Bit	IEEE 488 mandated
Status Byte Registers	Status Byte Register Service Request Enable Register	8 Bit	IEEE 488 mandated

Example for register use:

How to get this information: Is axis #1 now moving ?

Read the *Motion Condition Register* and analyze bit 1. If this bit is set, axis #1 is moving (actually the trajectory generator still outputs new profile positions).

Command: "STAT:MOT:COND?"

How to get this information: Has axis #1 moved since the last inquiry ?

Read the *Motion Event Register* and analyze bit 1. If this bit is set, axis #1 has changed its moving status since the last reading.

Command: "STAT:MOT:EVEN?" or "MOT?".

How to get this information: Which limit switches are hit.

Read the *Motion Condition Register* and analyze bits 8 to 15. Each of these 8 bits corresponds with either positive or negative limit switch of one or the 4 axes.

Command: "STAT:MOT:COND?", or "MOT:COND?".

How to get this information: Which limit switches were hit and automatic recovery was performed ?

Read the *Motion Event Register* and analyze bits 8-15.

Command: "STAT:MOT:EVEN?" or "MOT?".

4 Macro Programming

Since firmware release 2.0 macro programming is available. This feature allows to define command sequences stored permanently in the non volatile memory. Each macro command can be called up by an user defined individual name.

4.1 Non Volatile Macro Storage

Up to 16 macro programs can be stored in the non volatile RAM. Once stored, the command sequences keep permanently available also after a restart or power-up of the C-844.

Each macro program can be 500 Bytes long. The number of bytes used to store one command is composed of two bytes for the command-ID and 5 bytes for each parameter, e.g. the command "SOUR:TARG 25000" requires 7 bytes.

4.2 Macro Definition

To define a macro command sequence, fist the programming mode has to be activated by the command **"PROGram:DEFine:BEGin NAME"**, where NAME is a user selectable name with a maximum of 8 characters.

During the programming mode commands are not executed but stored in the macro storage.

The programming mode is terminated by the command **"PROGram:DEFine:END"**

Note: *Macro Buffer Overflow Handling :*
If the 500 bytes of macro storage are filled up and the new command does not fit anymore in the storage, this command is not stored and the programming mode is terminated automatically. A system error is written to the error queue and the LED at the front panel shines red.
Be aware that the next command then is executed immediately because the system is no longer in the programming mode.

Example:

```
PROG:DEF:BEG macro1
TARG:RPOS 5000
*WAI
PROG:DEF:END

PROG:DEF:BEG macro2
TARG:RPOS -2500
*WAI
PROG:DEF:END

PROG:DEF:BEG macro3
PROG:EXEC macro1
PROG:EXEC macro2
PROG:DEF:END
```

Macro *macro3* calls macros #1 and #2 for execution.

While in macro definition mode, these commands can not be used:

PROG:DEF	defining a macro
PROG:DEL	delete a macro
PROG:CAT?	Show all macro names stored
PROG:STAT	Show macro status
PROG:LOAD	Load macro
PROG:LOAD:COMM?	Show macro contents

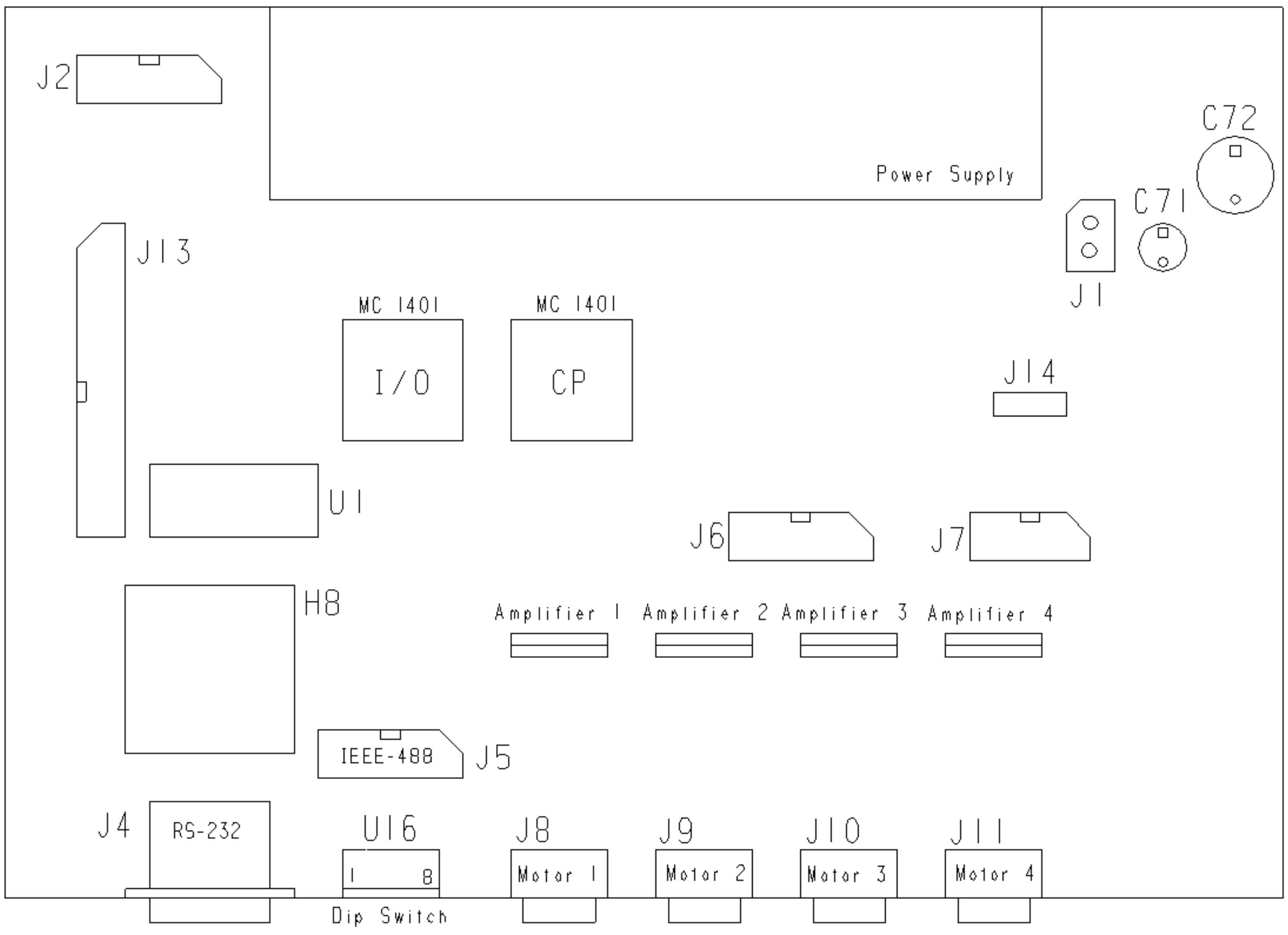
4.3 Starting Macros

A defined macro can be started by two different commands:

COMMAND	FUNCTION
PROGram:EXECute *name*	Direct option to start a macro. The macro runs one time and can not be stopped nor started during execution. The command can be used to run nested macros up to 5 levels deep. If no macro is defined under the specified name, an error message is generated.
PROG:LOAD name PROG:RUN n	This program sequence allows to run a macro n-times. The macro can be stopped and continued by the commands: PROG:STOP PROG:CONT Nested macros started by the PROG:EXEC command can be stopped and continued.

5 Hardware Architecture

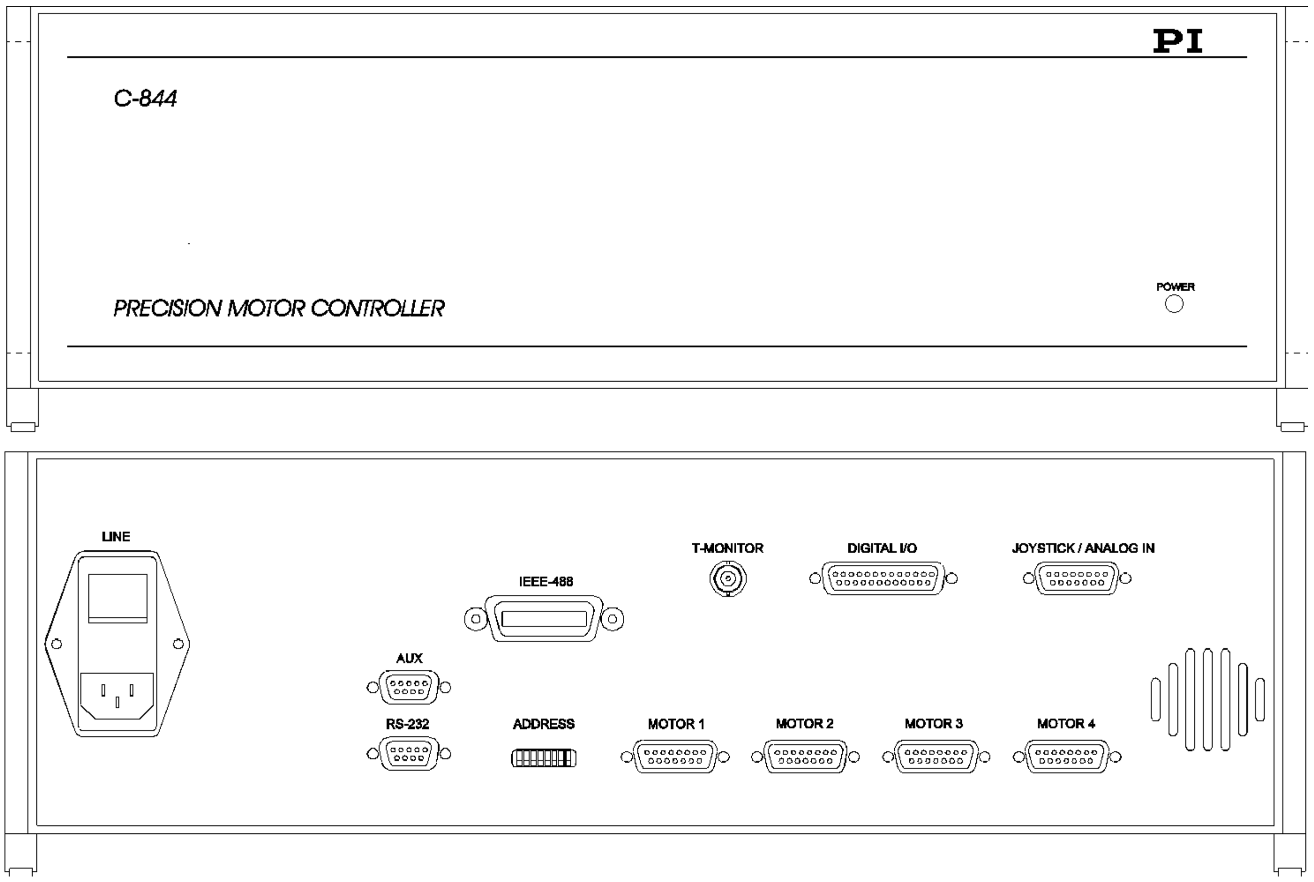
5.1 PCB Component Location



Dwg.: C844BLK1.HGL

J1	Power Supply input
J3	RS-232
J4	Auxiliary RS-232
J5	IEEE-488
J6	Digital I/O
J7	Analog I/O (optional analog input)

5.2 Front and Rear Panels



Dwg. C84400.WMF

5.2.1 Rear Panel Connectors and Controls

Label	Control Type	Function
LINE	Line power connector and switch	Line power receptacle
RS-232	DB9(m)	serial communication port
AUX	DB9(m)	2nd serial port
IEEE 488	IEEE 488 connector	IEEE 488 communication port
T-Monitor	BNC connector	Trajectory control output
Motor 1	DB15(f)	Motor connector for axis #1
Motor 2	DB15(f)	Motor connector for axis #2
Motor 3	DB15(f)	Motor connector for axis #3
Motor 4	DB15(f)	Motor connector for axis #4
ADDRESS	8 bit Dip switch	IEEE488 Address setting
DIGITAL I/O	DB25(f)	Digital Input and output lines
JOYSTICK IN (ANALOG IN)	DB15(f)	Analog input (joystick function not available)

5.3 Trajectory Monitor (T-Monitor)

The C-844 offers an analog trajectory monitor output for external control of the motor velocity. The incremental position encoder frequency is converted into an analog voltage within 0 to +10 Volts and applied to the BNC T-Monitor output.

Oscilloscope scans of the motor velocity over time indicated the settling behavior and help to fine adjust p-i-d-vff parameters.

5.4 Firmware Update

Since version 3.0 of the C-844 firmware, firmware updates can be accomplished via the serial communication port. and an external download program that transfers the firmware data file to the EEPROM.

The data file in conjunction with the download program is available on the PI-Motion CD (version 7 or later).

The download program (DOS) can be run under a DOS booted PC or under Windows 98. Some Windows NT systems may have problems to execute the program, in this case use a DOS bootable PC.

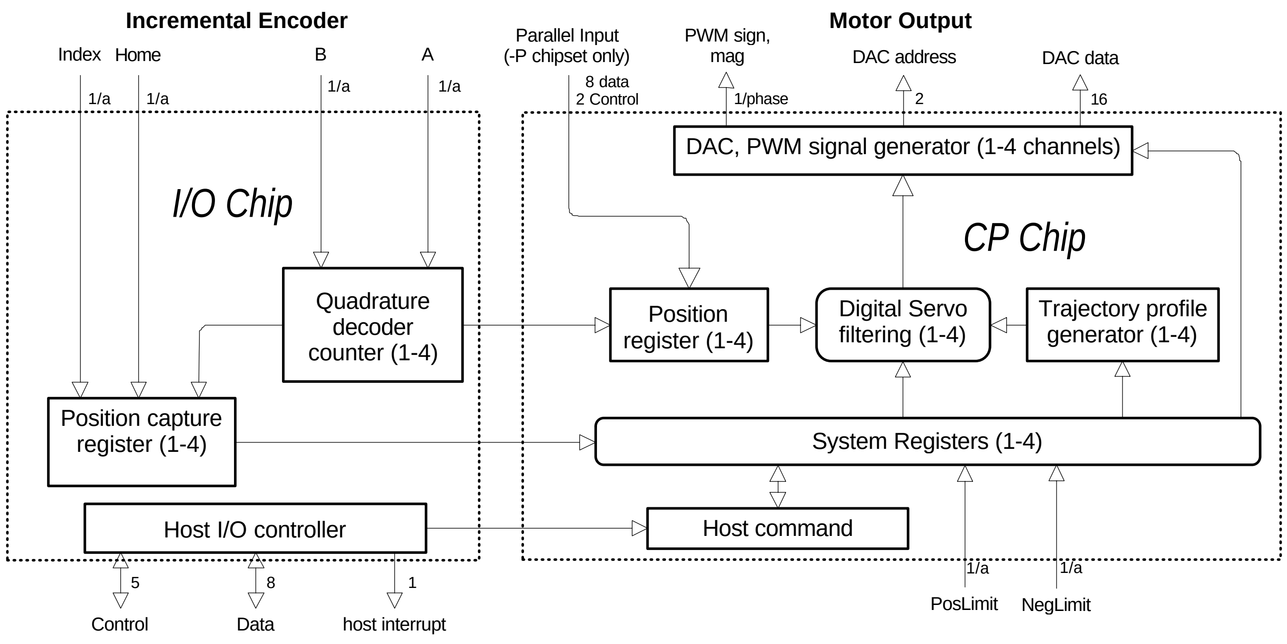
CRC data verification procedures during the download process offer the maximum of data integrity and continuous information of the download process.

Download Procedure

1. Connect the "AUX" connector of the C-844 by a serial communication cable (order number C-815.34, null-modem cable) with the COM2 port of the PC.
2. Turn C-844 power OFF
3. Boot the PC in the DOS mode or close all applications when running Windows.
4. Start the loader program "C844_upd.EXE" with the update version as parameter. Example: "C844_upd 5.10".
5. Wait until data file is loaded and "CONNECT" sign is blinking.
6. Switch C-844 power ON.
   The download process is now started and the front panel LED blinks green. If no corresponding data file is found, the program is terminated and an error message is displayed. During the download process the, the current mode is displayed (transmit, erase, store, CRC check). If any error during the transfer process occurs, the error code is displayed and the data field is send again. After all data fields are transferred successfully, again a CRC verification of the total firmware is performed. The succeeded verification terminated the download procedure with the message "FIRMWARE UPDATE OK".
7. If the firmware update is terminated before completion, the C-844 has no functional firmware stored and the LED blinks orange/green. In this case the entire update procedure has to be restarted.

6 Theory of Operations

Internal Block Diagram



The above figure shows an internal block diagram for the motion MC1401A control chipset used in the C-844.

Each servo axis inputs the actual location of the axis using incremental encoder signals. The incoming A, and B quadrature data stream is digitally filtered, and then passed on to a high speed up/down counter. This position information is then used to maintain a 32-bit actual axis position counter.

The chipset also supports the ability to capture the instantaneous position of each axis using an external trigger signal. The captured value may then later be retrieved by the host processor.

The chipset can be operated in two modes. Closed loop mode, which is the normal operating mode of the chipset, performs trajectory generation and digital servo loop closure. In this mode the motor output value is controlled by the servo filter. Open loop mode, which is used for direct motor-control operations only, does not use the output of the servo filter, and allows the motor output value to be controlled directly by the host processor.

When closed loop mode operations are used the actual axis position is combined with the target position generated by the trajectory profile generator to calculate a position error, which is passed through a PID filter. The resultant value is then output by the chipset to an external amplifier using either PWM or DAC signals.

6.1 Trajectory Profile Generation

The trajectory profile generator performs calculations to determine the target position, velocity and acceleration at each servo loop. These calculations are performed taking into account the current profile mode, as well as the current profile parameters set by the host. Four trajectory profile modes are supported:

- - S-curve point to point
- - Trapezoidal point to point
- - Velocity contouring
- - Electronic Gear

These commands select the profile mode :

```
[SOURce][:PROFile]:MODe SCURve
[SOURce][:PROFile]:MODe TRAPezoidal
[SOURce][:PROFile]:MODe VELocity
[SOURce][:PROFile]:MODe EGEar
```

The profile mode may be programmed independently for each axis. For example axis #1 may be in trapezoidal point to point mode while axis #2 is in S-curve point to point.

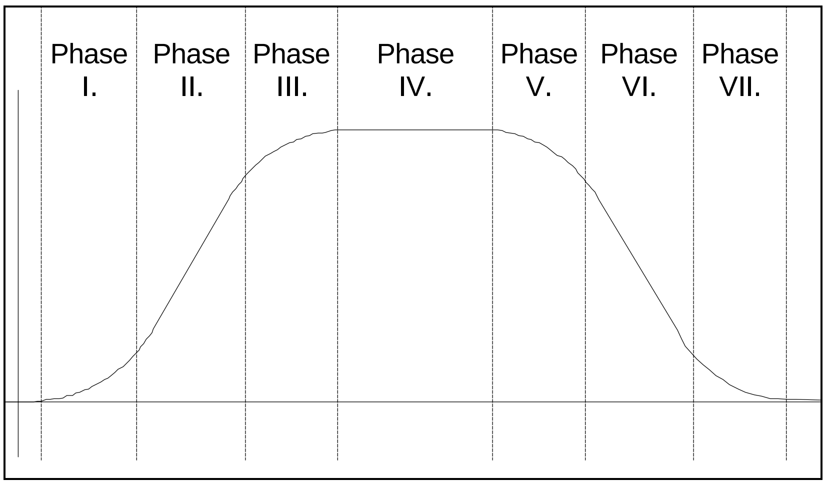
The appropriate mode is selected automatically if the current mode does not match with the move command:

`TARG:POS n`	Sets the trapezoidal mode if a non-point-to-point mode is active. If the S-curve mode is active, it stays active.
`TARG:VEL n`	Sets the velocity contouring mode
`TARG GRAT n`	Sets the electronic gear mode

6.1.1 S-Curve Mode

Use the following figure showing a typical S-curve velocity vs. time graph for reference in reading the next section:

S-Curve Profile

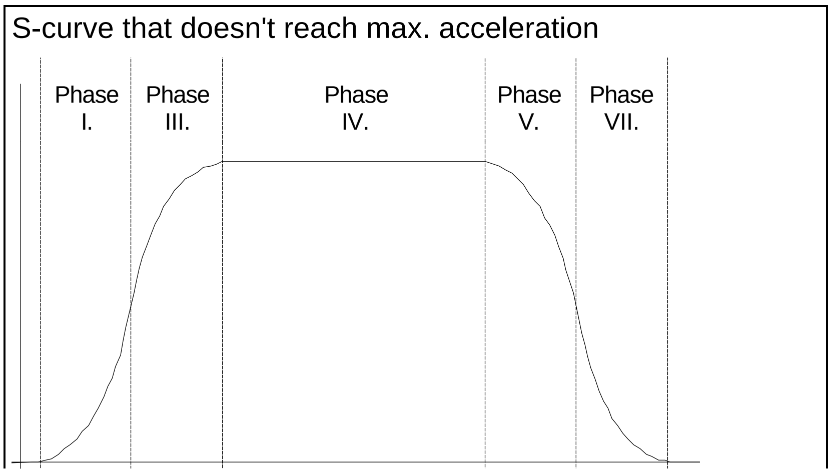


Velocity over Time

The S-curve profile drives the axis at the specified jerk until the maximum acceleration is reached. (phase I). It will then drive the axis at jerk = 0 (constant acceleration) through phase II. It will then drive the axis at the negative of the specified jerk though phase III, such that the axis reaches the specified maximum velocity with acceleration = 0. This completes the acceleration phase. At the end of the acceleration phase of the move, the velocity will be constant, and the acceleration will be 0. At the appropriate time, the profile will then decelerate (phases V, VI and VII) symmetrically to the acceleration phase such that it arrives at the destination position with acceleration and velocity = 0.

If a motion is commanded ("TARG n") during the fly, the axis is stopped and then the new move is performed towards the desired target is started. Note that the trapezoidal mode allows to overwrite target positions on the fly without stopping the motor.

There are several conditions where the actual velocity graph of an S-curve motion will not contain all of the segments shown in the above figure. For example, if the max. acceleration is not reached before the "half-way" point to the max. velocity, then the actual velocity profile will not contain a phase II or a phase VI segment (they will have a duration of 0 servo loops). Such a profile is shown below:



Another such condition is if the position is specified such that max. velocity is not reached. In this case there will be no phase IV, and there may also be no phase II and VI, depending on where the profile is "truncated".

Commands configuring the S-curve mode:

```
SCUR:JERK n
SCUR:MACC n
SCUR:MVEL n
```

6.1.2 Trapezoidal Mode

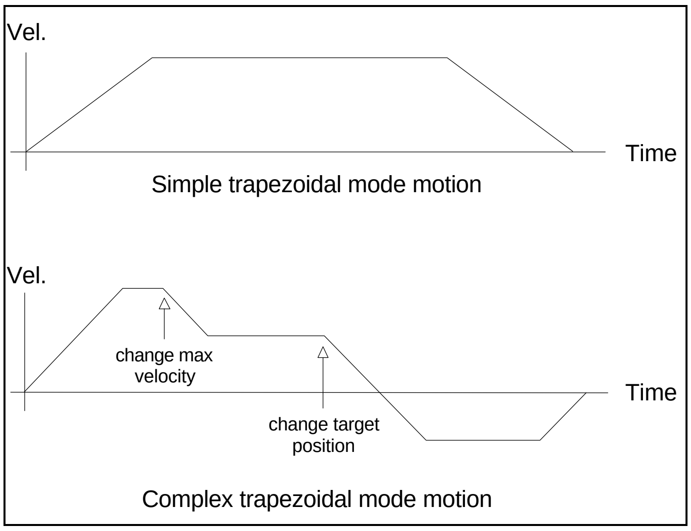
In the trapezoidal point to point profile mode the host specifies a destination position, a maximum velocity, and an acceleration. The trajectory is executed by accelerating at the commanded acceleration to the maximum velocity where it coasts until decelerating such that the destination position is reached with the axis at rest (zero velocity). If it is not possible to reach the maximum velocity (because deceleration must begin) then the velocity profile will have no "coasting" phase. The acceleration rate is the same as the deceleration rate.

If a new position is commanded ("TARG n") during the fly, the axis is not stopped. Depending on whether the new target is before or behind the current position, the motor continues to move or changes its turning direction with the programmed acceleration. In neither case the motion is interrupted (as it is in the S-curve mode).

Also a new maximum velocity can be specified while the axis is in motion. When this occurs the axis will accelerate or decelerate until the new velocity is reached.

Commands configuring the trapezoidal mode:

```
[SOUR:PROF:TRAP:]MVEL n
[SOUR:PROF:TRAP:]ACC n
```



The figure shows a velocity profile for a typical point to point trapezoidal move, along with a more complicated move involving on the fly changes to the maximum velocity and the destination position.

6.1.3 Velocity Contouring

In this profile mode the host specifies two parameters, the commanded acceleration, and the target velocity. The trajectory is executed by continuously accelerating the axis at the commanded rate until the target velocity is reached.

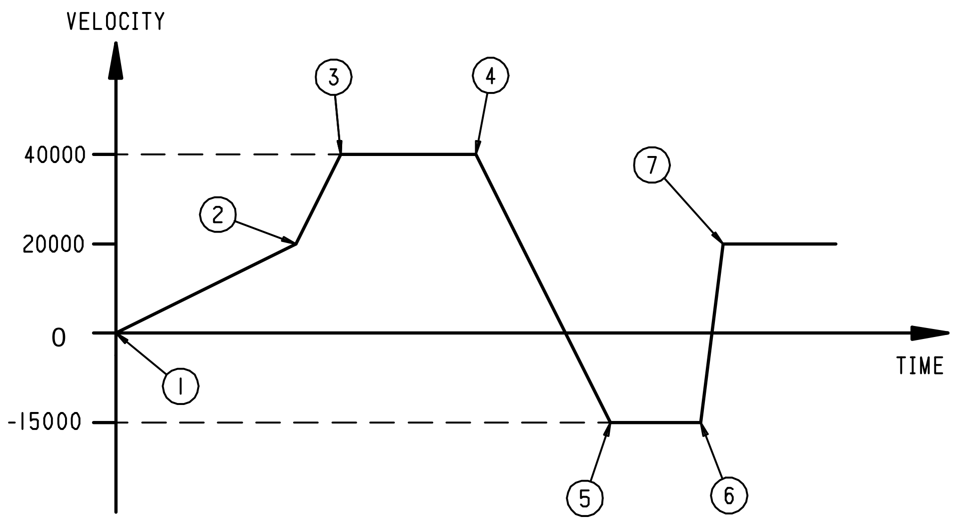
There are no restrictions on switching the profile mode to velocity contouring while the axis is in motion.

There are no restrictions on changing the profile parameters on the fly. Note that the motion is not bounded by position however. It is the responsibility of the host to generate acceleration and target velocity command values that allow safe motions within acceptable position limits.

The following figure shows a typical velocity profile using this mode.

Example of Velocity Contouring Mode

Velocity Profile over Time



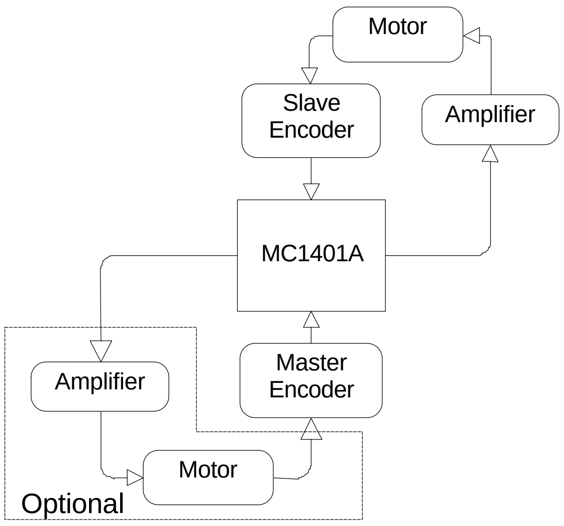
Dwg:A-MS052-001.HGL

(1) Start motion with "TARG:VEL 40000"
(2) Increase acceleration with "VEL:ACC n"
(3) Target velocity reached, now moves with constant velocity of 40000 c/s
(4) Program a new velocity with negative value that changes the turning direction with the command "TARG:VEL -15000"
(5) Target velocity reached, now moves with constant velocity of -15000 c/s
(6) Change target velocity and acceleration: "VEL:ACC n, TARG:VEL 20000"
(7) Target velocity reached, now moves with constant velocity of 20000 c/s

6.1.4 Electronic Gear

In this profile mode, the host specifies one parameter, the gear ratio. The target position is generated by applying the specified gear ratio to the current position of another axis, slaving the driven axis to the axis providing the position input. The following figure shows the arrangement for encoders and motor drives in a typical electronic gearing application.

Because a geared axis takes up two encoder channels, the total number of geared axes supported per chipset is 1/2 the total # of axes. In addition, the master /slave axis combinations are fixed.



The following chart shows the allowed master/slave combinations for each chipset:

Chipset p/n	gear pairs (master -> slave)
MC1401A	#3 -> #1, #4 -> #2
MC1201A	#2 -> #1
MC1101A	not available

Typically the master axis is only used for encoder input. It is possible however to use the master axis as a normal driven axis by leaving it enabled, and using one of the three trajectory modes other than electronic gear for the master axis. The net effect of this will be to run two servo motors off of the same trajectory profile (although at a different ratio if so programmed).

This configuration is shown in the previous diagram as 'optional' components. Using this configuration the chipset can be made to perform useful functions such as linear interpolation of two axis.

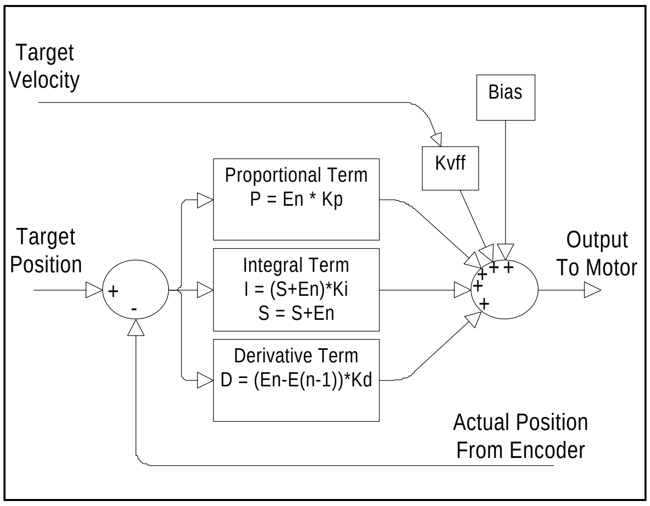
There are no restrictions on changing the gear ratio when the axis is in motion, although care should be taken to select ratios such that safe motion is maintained.

There are also no restrictions on changing to this profile mode while the axes is in motion.

6.2 Digital Servo Filtering

A digital filter is available for use in calculating a motor output signal. The filter used is a PID (proportional, integral, derivative) filter, along with a velocity feed-forward term and a term to adjust the offset, also called the DC bias value. This filter type is known as a PID+Vff filter.

This filter uses programmable gain values which can be tuned to provide excellent control accuracy and stability over a large range of systems.


Target Velocity
Bias
Kvff
Proportional Term P = En * Kp
Integral Term I = (S+En)*Ki S = S+En
Derivative Term D = (En-E(n-1))*Kd
Target Position
Output To Motor
Actual Position From Encoder

The schematic diagram shows the computational flow for the PID+Vff digital filter.

In the PID+Vff filter, the host-specified parameters are:

Kp:	Proportional Gain
Ki:	Integral Gain
Kd:	Derivative Gain
ilim:	Integration limit
Kvff:	Velo.feedforward gain
MtrBias:	DC motor offset

Commands to figure the digital output filter:

```
[SOUR:][FILT:][GAIN:]PID p,i,d
[SOUR:][FILT:][GAIN:]VFF n
[SOUR:][FILT:]BIAS n
[SOUR:][FILT:]LIM:IERR n
[SOUR:][FILT:]LIM:PERR n
```

7 SCPI Commands

SCPI means "Standard Commands for Programmable Instruments".

The functionality of the C-844 is reflected by the hierarchically structured set of commands based on the SCPI format.

The goal of SCPI is to reduce program developing time. SCPI accomplishes this goal by providing a consistent programming command environment for instrument control and data usage.

SCPI offers numerous advantages to the application engineer. Due to the structured command set, die user easily recognises the logic behind the commands and helps the user to memorise the command names. Defined program messages and data formats across all SCPI instruments, regardless of manufacturer, make the C-844 familiar to the user.

Compared with other, non structured command sets, SCPI command strings are longer but easier to interpret due to the tree structure. Also, many nodes are optional and do not need to be included, others can be abbreviated by the first 3 or 4 characters. Using all shortening options, also SCPI commands can be very compact.

Command keywords can be used in a long form (ACCeleration) or short form shown in capital letters (ACC).

C-844 supports full SCPI format with optional nodes, standard abbreviations and IEEE488 mandated commands.

7.1 General Rules for SCPI

- Both long and short form of the commands can be used and also a mixed syntax.
- Command keywords are separated by a ":".
- Parameters are separated from the command queue by a space and may consist of a number of the specified type or a keyword like "ON" or "OFF". Options available are defined in the SCPI Command Reference section.
- All set-commands can also be used as queries when terminated with a "?".
- More than one command can be compound in one command line. Individual commands are separated by ";".

Example :

Motor #2 shall perform a relative move for 5000 counts. If axis #2 is not active, it has to be activated using the AXIS:SELECT command.

Then the move command follows:

Format :	`[SOURce:]TARGet:[LEVel:][IMMediate:]RPOSition 5000`
Form a)	`SOURCE:TARGET:LEVEL:IMMEDIATE:RPOSITION 5000`
Form b)	`SOUR:TARG:LEV:IMM:RPOS 5000`
Form c)	`TARG:RPOS 5000`

7.2 Command Examples

7.2.1 Sequential Commands

All commands are sequential commands except those with a pending flag. Sequential commands are executed and terminated immediately and need no *time* for execution, e.g. setting the filter bias to a new value is executed at once.

7.2.2 Overlapping Commands

Overlapping commands are those who require a certain time for execution, e.g. move commands need some time until the desired target is reached and the command is terminated.

Any new move command would move the target to a new position and the old one will never be reached. To avoid this, a command in-execution flag is set during the execution of an overlapped command.

Also the velocity in the velocity contouring mode is an overlapping command.

Overlapping commands can be sequenced by using the *WAI command. If any command in execution has the overlapping flag set, the *WAI command holds the next command until all overlapping commands are terminated. This allows to send the next move command before the current set target is reached without aborting the current move.

Examples:

Commands	Action
TARG:POS 25000 TARG:POS 15000	Motor starts at 0 and moves to position 15000. The previous target of 25000 is not reached because it is overwritten by next command
TARG:POS 25000 *WAI TARG:POS 15000	Motor starts at 0 and moves to position 25000. The *WAI command enforces all overlapped commands to be completed before the following command is executed. After the position 25000 has been reached, the motor moves back to position 15000.
AXIS:POS 0 TARG:POS 5000 *WAI TARG:POS -10000 *WAI TARG:POS 0	After the position counter is cleared, the motor is moved to position 5000, then to -10000 and at last back to 0.
AXIS:POS 50000 AXIS 1 TARG:RPOS 20000 *WAI AXIS 2 TARG:RPOS 20000 *WAI TARG HOME AXIS 1 TARG HOME	: set all position counters to 50000 : select axis 1 : move axis 1 for +20000 counts : wait until move command is completed : select axis 2 : move axis 2 for +20000 counts : wait until move is completed : move axis 2 to zero position : select axis 1 : move axis 1 to zero position

7.3 IEEE488.2 mandated Commands

C-844 Motor Controllers conform to the IEEE488.2 specifications for devices. All common commands declared mandatory by IEEE 488.2 are implemented.

Mnemonic	Name
*RST	Performs a power-on reset
*IDN?	Reads the device identification string
*STB?	Reads the status byte
*CLS	Clears status/condition, event register and error queue.
*ESE	Sets the standard event enable mask. This mask defines which of the standard events set in the *standard event status register* causes the SESR summary bit to be set in the status byte. Range of the parameter: 0..255
*ESE?	Reads the mask of the standard event status register.
*ESR?	Reads and clears the standard event status register.
*OPC	Waits until all overlapped commands are terminated. Then sets the operation complete bits in the standard event status register.
*OPC?	Works like the *OPC command and waits until all overlapped commands are executed. Unlike *OPC it sets not a bit but place a "1" in the output queue (asynchron device response).
*SRE	Sets the service request mask. This mask determines which bits of the status byte will set the RQS bit. Range of the parameter: 0..255
*SRE?	Reads the service request enable mask.
*TST?	Starts a self test routine and reports legal positions and motion errors.
*WAI	Wait-to-continue command. It enforces a sequential execution of overlapped commands. New commands are not executed until the last command is terminated. During that time all new commands are hold in the command queue.

8 C-844 SCPI Command Set

8.1 Command Survey

Branch: AXIS

AXIS

[:SELect]	*Select active axis*
:POSition	*Defines / reads the current encoder position*
:STATe	*Enables / disables axis amplifiers*
:LIMit	
[:STATe]	*Enables / disables limit switch handling*
:SENSe	*Defines limit switch signal levels*
:TERMinator	*Chooses pull-up/down termination for LS lines*
:LEVel?	*Reads state of limit switches*
:BRAKe	*Enables / disables motor brake*
:SBRAke	*Defines power on default brake setting*
:ENCoder	*Choose encoder type: differential / single ended*

Branch: SOURCE

[FILTer:]

BIAS	*Set digital filter bias*
[GAIN:]	
VFForward	*Set velocity feed forward gain*
PID	*Set p-i-d- parameters*
PERRor?	*Read current position error*
IERRor?	*Read current integration error*
LIMit:	
PERRor	*Sets max. position error allowed*
IERRor	*Sets max. integration error allowed*

Branch: OUTPUT

OUTPut: *Sets motor voltage in servo off*

SERVo	*Enables / disables servo mode*
SIGNal	*Sets PWM/DAC*
LIMit	*Limits max. motor voltage*

Branch PORT

[PORT:]

DINput?	*Reads the digital input port*
DOUTput	*Writes to the digital output port*
TRIGger:	
DIN1	*Defines trigger action*
DIN2	*Defines trigger action*
DIN3	*Defines trigger action*

DIN4	*Defines trigger action*
DIN5	*Defines trigger action*
DIN6	*Defines trigger action*
DIN7	*Defines trigger action*
DIN8	*Defines trigger action*
DOUT	*Defines trigger output*

Branch: SOURCE

[PROFile:]

MODe	*Set Mode*
[TRAPezoidal:]	
MVELocity	*Max. velocity for trapez mode*
ACCeleration	*Acceleration for trapez mode*
SCURve:	
MVELocity	*Max. velocity for S-curve mode*
MACCeleration	*Max. acceleration for S-curve mode*
JERK	*Set jerk*
VELocity:	
ACCeleration	*Acceleration for velocity mode*
CURRent:	
TPOSition?	*reads the profile target*
TVELocity?	*reads the profile target velocity*
STOP	*Immediate motor stop of all axes*
HALT	*Stop motion, smoothly*

Branch PROGRAM

PROGram:

EXECute	Start a macro command
LOAD	*Loads a macro program for execution*
:COMMands	*Reads the macro command sequence*
[STATe:]	*Controlls the active macro program*
RUN	*Starts a macro*
CONTinue	*Continues a stopped macro*
STOP	*Breaks a macro*
[MAINtain:]	
DEFine:	
BEGin	*Starts a macro definition*
END	*Terminates a macro definition*
DELete	*Erases a macro storage*
CATalog?	*Reads all registered macro names*
AUTO	*Start a macro at power on*

Branch: STATUS

[STATus:]

MOTion	
[:EVENt]?	*Reads the motion event register*
:CONDition?	*Reads the motion condition register*

	:ENABle		*Sets the motion enable mask*
OPERation			
	[:EVENt]?		*Reads the operation event register*
	:CONDition?		*Reads the operation condition register*
	:ENABle		*Sets the operation enable mask*
QUEStionable			
	[:EVENt]?		*Reads the questionable event register*
	:CONDition?		*Reads the questionable condition register*
	:ENABle		*Sets the questionable enable mask*
PRESet			*Resets enable masks*

Branch: SYSTEM

[SYSTem:]

ERRor			
	[:NEXT]?		*Reads device error number*
PASSword			*Defines a password*
	[:CENable]		*Enables password protection*
		:STATe?	*Reads password status*
	:CDISable		*Disables password protection*
DEVice			*Writes device information, password protected*
WAIT			*Wait with command execution for n milliseconds*

Branch SOURCE

TARGet

[:IMMediate]		
	[:POSition]	*Absolute Position*
	:RPOSition	*Relative Position*
	:VELocity	*Absolute velocity*
	:RVELocity	*Relative velocity*
	:GRATio	*Electronic gear ratio*
	:RGRatio	*Relative electronic gear ratio*
:TRIGger		
	:POSition	define trigger position
	:VELocity	define trigger velocity
	:GRATio	define trigger gear ratio
:LIMit		
	:NPOSition	*Negative position*
	:PPOSition	*Positive position*
	:VELocity	*v-limit for velocity mode*
	:GRATio	*Limit for electronic gear ratio*
	:STATe	*Limit status*
:FIND	[:REFerence]	search origin position

8.2 Command Reference

8.2.1 *Moving Targets*

TARG {<value>|HOM|CURR|PLIM|NLIM|REF}

Function:	Commanding a new absolute motor position in counts. The command applies to the active axis. Parameters of the current active mode are used (MVel, MAcc, Acc, Jerk) Logical positions like positive or negative limits (PLIM, NLIM), home (HOM), reference (REF) can be used alternatively to numeral values.
Notes:	Switches to trapezoidal mode if a non point-to-point mode is active.
Examples:	Long form: **SOURce:TARGet:LEVel:IMMediate:POSition 23500** Short form: **TARG 23500**
CrossRef:	MC1401: SET_POS ShortCommand: **MA** (Move Absolute)

TARG?

Function:	Query of the commanded target position. The command reads the active axis.
Examples:	Long form: **SOURce:TARGet:LEVel:IMMediate:POSition?** Short form: **TARG?**
CrossRef:	MC1401: GET_POS ShortCommand: **TT** (Tell Target)

TARG:RPOS <value>

Function:	Commanding a relative move starting at the current position.
Notes:	Overlapped command, can be sequenced by *WAI.
Examples:	Long form: **SOURce:TARGet:LEVel:IMMediate:RPOS 5000** Short form: **TARG:RPOS 5000**
CrossRef:	MC1401: not available ShortCommand: **MR** (Move Relative)

TARG:RPOS?

Function:	Reads the last commanded value for relative move.
Examples:	Long form: **SOURce:TARGet:LEVel:IMMediate:RPOSition?** Short form: **TARG:RPOS?**
CrossRef:	MC1401: not available ShortCommand: **MR?**

TARG:VEL {<value>|CURR}

Function:	Commanding a velocity in counts/s. The profile mode is set to velocity and the motor starts in that direction according to the sign of the velocity value.
Range:	-16383..+16383
Notes:	Overlapped command, can be sequenced by *WAI.
Examples:	Long form: **SOURce:TARGet:LEVel:IMMediate:VEL 12000** Short form: **TARG:VEL 12000** **TARG:VEL -12000**
CrossRef:	MC1401: SET_VEL ShortCommand: **SVV** (Set Velocity of Velocity mode)

TARG:VEL?

Function:	Reads the last commanded velocity
Examples:	Long form: **SOURce:TARGet:LEVel:IMMediate:VELocity?** Short form: **TARG:VEL?**
CrossRef:	MC1401: GET_VEL ShortCommand: **SVV?**

TARG:RVEL <value>

Function:	Commanding a new velocity *relative* to the current velocity in counts/s. The profile mode is set to velocity and the motor starts or stays moving accelerating/decelerating to the new velocity.
Range:	0..+16383
Notes:	Overlapped command, can be sequenced by *WAI.
Examples:	Long form: SOURce:TARGet:LEVel:IMMediate:RVEL 1000 Short form: TARG:RVEL 1000
CrossRef:	MC1401: not available ShortCommand: **IVV** (Increment Velocity for velocity mode)

TARG:RVEL?

Function:	Reads the last by IVV commanded relative velocity
Examples:	Long form: **SOURce:TARGet:LEVel:IMMediate:RVELocity?** Short form: **TARG:VEL?**
CrossRef:	MC1401: not available ShortCommand: not available

TARG:GRAT <value>

Function:	Commanding an electronic gear ratio for the active axis, which is either axis #1 or #2 (slave axes). The master axes are #3 (master for axis #1) and #4 (axis for axis #2). The command is rejected if none of axes #1 or #2 are selected.

	The axis is switched immediately to electronic gear mode.
Range:	-32768..+32767
Examples:	Long form: SOURce:TARGet:LEVel:IMMediate:GRATio 2000 Short form: TARG:GRAT 2000
CrossRef:	MC1401: SET_Ratio ShortCommand: **SRA** (Set Ratio)

TARG:GRAT?

Function:	Reads the current gear ratio. Also relative gear ratio settings are taken into account.
Examples:	Long form: **SOURce:TARGet:LEVel:IMMediate:GRAT?** Short form: **TARG:GRAT?**
CrossRef:	MC1401: GET_RATIO ShortCommand: **SRA?**

TARG:RGRAT <value>

Function:	Commanding a relative electronic gear ratio for the active axis, which can be either axis #1 or #2 (slave axes). The master axes are #3 (master for axis #1) and #4 (axis for axis #2). The command is rejected if none of axes #1 or #2 are selected. The axis is switched immediately to electronic gear mode and the RGRAT value is added to the old one.
Range:	-32768..+32767
Examples:	Long form: **SOURce:TARGet:LEVel:IMMediate:RGRATio 1500** Short form: **TARG:RGRAT 1500**
CrossRef:	MC1401: not available ShortCommand: not available

TARG:RGRAT?

Function:	Reads the last commanded relative gear ratio.
Examples:	Long form: **SOURce:TARGet:LEVel:IMMediate:RGRAT?** Short form: **TARG:RGRAT?**
CrossRef:	MC1401: not available ShortCommand: not available

TARG:FIND {POS | NEG | AUTO | BUTO}

Function: Start a search run for the reference signal (origin position of the stage). Can be used with stages having a reference sensor installed.

Parameter meanings:

POS:	start search in positive direction
NEG:	start search in negative direction
AUTO:	start search depending on current input level (high=POS)
BUTO	start search depending on current input level (high=NEG)

	AUTO | BUTO parameters can be used with stages having a side-coded reference signal option (like PI M-500 stages). In this case the stage starts automatically in the right direction towards the reference point.
Examples:	Long form: **SOURce:TARGet:FIND:REFerence POSitive** Short form: **TARG:FIND POS**
CrossRef:	MC1401: not available ShortCommand: **FEP**, **FEN** (Find Edge Positive, Negative)

Note:

The auto search functions reach the reference edge from an undefined direction. Due to the magnetic hysteresis behavior of the reference sensors, the reference sensor actually has two edges, depending on the direction from where it is approached. In order to make sure to approach the reference point every time from the same direction, also with the auto search modes a second approach from the desired direction may be required.

TARG:FIND?

Function:	Reads the current status of the reference sensors, whether the reference position was found or not.
Examples:	Long form: **SOURce:TARGet:FIND:REFerence?** Short form: **TARG:FIND:REF?**
Report:	"ON" : Reference found "OFF" : Reference not found
CrossRef:	MC1401: not available

8.2.2 *Axis Commands*

AXIS {1|2|3|4}

Function:	This command selects the active axis. Also the velocity to analog signal of the active axis is output at the T-MONITOR output BNC-connector.
Examples:	Long form: **AXIS:SELECT 2** Short form: **AXIS 2**
CrossRef:	MC1401: SET_1, SET_2, SET_3, SET_4 ShortCommand: not required, axis selection done by commands

AXIS?

Function:	Reads the current activated axis.
Examples:	Long form: **AXIS:SELect?** Short form: **AXIS?**
CrossRef:	MC1401: not available

AXIS:STAT {ON|OFF}

Function:	This command enables or disables the axis. If the axis is in the OFF state, the amplifier is shut off. The axis can be configured by commands and these settings will become effective when the axis is set in the ON state.
Examples:	Long form: **AXIS:STATe ON** Short form: **AXIS:STAT ON**
CrossRef:	MC1401: AXIS_ON, AXIS_OFF ShortCommands: **AN, AF** Axis ON, Axis OFF)(

AXIS:STAT?

Function:	Reports the axis status.
Examples:	Long form: **AXIS:STATe?** Short form: **AXIS:STAT?**
CrossRef:	MC1401: GET_STATUS (Bits 4,5,6,7) ShortCommand: **TS** (Tell Status)

AXIS:BRAK {ON|OFF}

Function:	This command activates the brake for the active axis. Actually the output voltage for the motor brake is set. Also an integer can be used as parameter: 0=OFF, not 0 = ON.
Examples:	Long form: **AXIS:BRAKe ON** Short form: **AXIS:BRAK ON**

CrossRef:	MC1401: not available ShortCommand: **BK**{0|1}

AXIS:BRAK?

Function:	Reads the status of the motor brakes.
Examples:	Long form: **AXIS:BRAKE?** Short form: **AXIS:BRAK?**
CrossRef:	MC1401: not available ShortCommand: **BK?**

AXIS:SBRA {ON|OFF}

Function:	This command defines the motor brake state at power on. If a stage with additional mechanical load is mounted vertically, the stage brake must stay clamped when the C-844 is powered up. In this case the command AXIS:SBRA ON has to be send for the regarding axis.
Examples:	Long form: **AXIS:SBRAKE ON** Short form: **AXIS:SBRA ON**
CrossRef:	MC1401: not available

AXIS:SBRA?

Function:	Reads the status of the power on default brake setting.
Examples:	Long form: **AXIS:SBRAKE?** Short form: **AXIS:SBRA?**
CrossRef:	MC1401: not available

8.2.3 *Position / Encoder Handling*

AXIS:ENC {SEND|DIFF}

Function:	This command defines the encoder signal detection mode. Both single ended (**SENDed**) and differential (**DIFFerential**) signals are used with PI mechanics. Axes #1,#2 and #3,#4 can only be set to the same mode.
Examples:	Long form: **AXIS:ENCoder DIFF** Short form: **AXIS:ENC DIFF**
CrossRef:	MC1401: not available

AXIS:ENC?

Function:	Reads the active encoder signal detection mode.

Examples:	Long form: **AXIS:ENCODER?** Short form: **AXIS:ENC?**
CrossRef:	MC1401: not available

AXIS:POS {<value>|HOME}

Function:	This command sets the current encoder count position to a new value. After power-on the counter is reset to zero. HOME is a logical position and equals 0 counts. Also during the axis is in motion AXIS:POS can be used. In this case the target position is adapted to the new scale.
Examples:	Long form: **AXIS:POSITION 60000** Short form: **AXIS:POS 60000**
CrossRef:	MC1401: SET_ACTL_POS, ZERO_POS ShortCommand: **SP** (Set Position) **DH** (Define Home)

AXIS:POS?

Function:	Reads the current encoder (motor) position.
Examples:	Long form: **AXIS:POSition?** Short form: **AXIS:POS?**
CrossRef:	MC1401: GET_ACTL_POS ShortCommand: **TP** (Tell Position)

8.2.4 *Limit Switch Handling*

AXIS:LIM {ON|OFF}

Function:	This command enables or disables the over travel limit switch control. As default, all limit switches are enables.
Examples:	Long form: **AXIS:LIMit:STATe ON** Short form: **AXIS:LIM ON**
CrossRef:	MC1401: LMTS_ON, LMTS_OFF ShortCommand: **LN, LF**

AXIS:LIM?

Function:	Reads the status of the limit switch control status.
Examples:	Long form: **AXIS:LIMit:STATe?** Short form: **AXIS:LIM?**
Report:	"ON" | "OFF"
CrossRef:	MC1401: not available

AXIS:LIM:SENS {HIGH|LOW}

Function:	This command defines the external signal level triggering the limit switch event. The level can be +5Volt or GND.
Examples:	Long form: **AXIS:LIMit:SENS HIGH** Short form: **AXIS:LIM:SENS HIGH**
Report:	"LOW" | "HIGH"
CrossRef:	MC1401: SET_LIM_SENS ShortCommand: **SLS**

AXIS:LIM:SENS?

Function:	Reads the level that the firmware interprets as "Limit Active".
Examples:	Long form: **AXIS:LIMit:SENS?** Short form: **AXIS:LIM:SENS?**
CrossRef:	MC1401: not available ShortCommand: **SLS?**

AXIS:LIM:TERM {PUPP|PDOW|Z|DEF}

Function:	This command defines the termination of the limit switch input lines inside the C-844. Possible options are PUPP=pull-up, PDOWn=pull-down, Z=tri-state and DEF=PUPP=pull-up.
Note:	Z-termination can only be applied to all 4 axes. After the command **AXIS:LIM:TERM Z** all 4 axis are Z-terminated. The next command like **AXIS:LIM:TERM PUPP** will set all axis to pull-up termination. Then also individual settings for PUPP, PDOW or DEF are possible.
Examples:	Long form: **AXIS:LIMit:TERMinator PUPP** Short form: **AXIS:LIM:TERM PUPP**
CrossRef:	MC1401: not available ShortCommand: **SLT {0|1}**

AXIS:LIM:TERM?

Function:	Reads the termination of the limit switch lines.
Examples:	Long form: **AXIS:LIMit:TERMination?** Short form: **AXIS:LIM:TERM?**
Report:	"PUPP"|"PWOW"|"Z"
CrossRef:	MC1401: not available ShortCommand: **SLT?**

AXIS:LIM:LEV?

Function:	Reads the physical voltage levels of the limit switch lines.
Examples:	Long form: **AXIS:LIMIT:LEVEL?** Short form: **AXIS:LIM:LEV?**
Report:	"N-Limitline HIGH, P-Limitline LOW"

CrossRef:	MC1401: GET_LIM_SWITCH ShortCommand: **GLS**

8.2.5 *Defining Limits*

TARG:LIM:NPOS <value>

Function:	This command defines a software limit for negative positions. It works for S-curve and trapezoidal modes. Any position commanded targeting to the negative side of the limit is refused.
Range:	> -1.073.741.824
Examples:	Long form: **SOURce:TARGet:LIMit:NPOSition -100000** Short form: **TARG:LIM:NPOS -100000**
CrossRef:	MC1401: not available ShortCmd: **SLN**

TARG:LIM:NPOS?

Function:	Reads the programmed value for negative software limit.
Examples:	Long form: **SOURce:TARGet:LIMit:NPOS?** Short form: **TARG:LIM:NPOS?**
CrossRef:	MC1401: not available ShortCmd: **SLN?**

TARG:LIM:PPOS <value>

Function:	This command defines a software limit for positive positions. It works for S-curve and trapezoidal modes. Any position commanded targeting to the positive side of the limit is refused.
Range:	< 1.073.741.824
Examples:	Long form: **SOURce:TARGet:LIMit:PPOSition 50000** Short form: **TARG:LIM:PPOS 50000**
CrossRef:	MC1401: not available ShortCmd: **SLP**

TARG:LIM:PPOS?

Function:	Reads the programmed value for positive software limit.
Examples:	Long form: **SOURce:TARGet:LIMit:PPOS?** Short form: **TARG:LIM:PPOS?**
CrossRef:	MC1401: not available ShortCmd: **SLP?**

TARG:LIM:VEL <value>

Function:	This command defines a software limit for maximum velocities in the velocity mode. Any velocity (absolute value) commanded larger than the limit is not applied and causes an error message.
Range:	value< 40.957.500
Examples:	Long form: **SOURce:TARGet:LIMit:VELocity 12000** Short form: **TARG:LIM:VEL 12000**
CrossRef:	MC1401: not available ShortCmd: **DLV**

TARG:LIM:VEL?

Function:	Reads the programmed velocity limit.
Examples:	Long form: **SOURce:TARGet:LIMit:VEL?** Short form: **TARG:LIM:VEL?**
CrossRef:	MC1401: not available ShortCmd: **DLV?**

TARG:LIM:GRAT <value>

Function:	This command defines a software limit for the gear ratio in the electronic gear mode. Any ratio commanded (absolute value) above the limit is refused and an error message is queued.
Range:	< 32.767
Examples:	Long form: **SOURce:TARGet:LIMit:GRATio 3500** Short form: **TARG:LIM:GRAT 3500**
CrossRef:	MC1401: not available

TARG:LIM:GRAT?

Function:	Reads the programmed limit for gear ratio.
Examples:	Long form: **SOURce:TARGet:LIMit:GRAT?** Short form: **TARG:LIM:GRAT?**
CrossRef:	MC1401: not available

TARG:LIM:STAT {ON|OFF|<value>}

Function:	This command enables or disables the programmed limits. Limits are taken into account in the ON status and are not used in the OFF status. Also integer numbers can be used as parameter: 0 equals OFF, any other number equals ON.
Examples:	Long form: **SOURce:TARGet:LIMit:STATus ON** Short form: **TARG:LIM:STAT ON**
CrossRef:	MC1401: not available

TARG:LIM:STAT?

Function:	Reads the limit status.

Examples:	Long form: **SOURce:TARGet:LIMit:STATte?** Short form: **TARG:LIM:STAT?**
CrossRef:	MC1401: not available

8.2.6 *Profile Generation*

STOP

Function:	Emergency break for all axes. An immediate motor stop of all axes is performed. The current target velocity is set to zero and the new target equals the current position. All commands, still pending or not yet completed are canceled. The *WAI-command is overwritten, so the STOP command can not be sequenced. If STOP is send while a compound sequence is active, all further not yet terminated commands are canceled. If STOP is send during a macro program execution, the macro is terminated and removed from the load storage. If any axis was set to the velocity or e-gear-mode, after the STOP these axes are set to the trapezoidal mode.
Note:	The STOP command works like an emergency stop-switch.
Examples:	Long form: **SOURce:PROFile:STOP** Short form: **STOP**
CrossRef:	MC1401: STOP ShortCmd: **AB** (Abort Motion)

HALT

Function:	This command causes the motor of the active axis to stop with programmed acceleration. It has the same functionality like the PROF:STOP command but is breaks the motor more smoothly. This command is not available in the electronic gear mode.
Examples:	Long form: **SOURce:PROFile:HALT** Short form: **HALT**
CrossRef:	MC1401: SMOOTH_STOP ShortCmd: **ST**

MOD {TRAPezoidal|SCURve|VELocity|EGEar}

Function:	This command defines the profile generation mode. Trapezoidal and S-curve mode can only be started if the axis is at rest. Electronic gear and velocity mode become effective immediately.
Examples:	Long form: **SOURce:PROFile:MODe TRAPezoidal** Short form: **PROF:MOD TRAP**

CrossRef:	MC1401:	SET_PRFL_TRAP, SET_PRFL_SCURVE, SET_PRFL_VEL, SET_PRFL_GEAR
	ShortCmd:	**SPM**{0|1|2|3}

MODe?

Function:	Reads the programmed profile mode
Examples:	Long form: **SOURce:PROFile:MODe?** Short form: **MOD?**
CrossRef:	MC1401: GET_MODE (Bit 11, 12) ShortCmd: **SPM?**

SCUR:MVEL <value>

Function:	This command sets the maximum velocity for S-curve profile mode.	
Range:	0.. 40.957.500 [c/s]	
Examples:	Long form:	**SOURce:PROFile:SCURve:MVEL 45000**
	Short form:	**SCUR:MVEL 45000**
CrossRef:	MC1401:	SET_VEL
	ShortCmd:	**SSV**

SCUR:MVEL?

Function:	Reads the programmed maximum velocity for S-curve mode.
Examples:	Long form: **SOURce:PROFile:SCURve:MVelocity?** Short form: **SCUR:MVEL?**
CrossRef:	MC1401: GET_VEL ShortCmd: **SSV?**

SCUR:MACC <value>

Function:	This command sets the maximum acceleration for S-curve profile mode.	
Range:	0.. 81.917.500 [c/s/s]	
Examples:	Long form:	**SOURce:PROFile:SCURve:MACCL 30000**
	Short form:	**SCUR:MACC 30000**
CrossRef:	MC1401:	SET_MAX_ACC
	ShortCmd:	**SSA**

SCUR:MACC?

Function:	Reads the programmed maximum acceleration for S-curve mode.
Examples:	Long form: **SOURce:PROFile:SCURve:MACCLeration?** Short form: **SCUR:MACC?**
CrossRef:	MC1401: GET_MAX_ACC ShortCmd: **SSA?**

SCUR:JERK <value>

Function:	This command sets the jerk for S-curve mode.	
Range:	[c/s/s/s]	
Examples:	Long form:	**SOURce:PROFile:SCURve:JERK 4500**
	Short form:	**SCUR:JERK 4500**
CrossRef:	MC1401:	SET_JERK
	ShortCmd:	**SSJ**

SCUR:JERK?

Function:	Reads the programmed jerk for S-curve mode.	
Examples:	Long form: **SOURce:PROFile:SCURve:JERK?**	
	Short form: **SCUR:JERK?**	
CrossRef:	MC1401:	GET_JERK
	ShortCmd:	**SSJ?**

MVEL <value>

Function:	This command sets the maximum velocity for the trapezoidal mode.	
Range:	0.. 40.957.500 [c/s]	
Default:	6000	
Examples:	Long form:	**SOURce:PROFile:TRAPezoidal:MVEL 120000**
	Short form:	**MVEL 120000**
CrossRef:	MC1401:	SET_VEL
	ShortCmd:	**SV**

MVEL?

Function:	Reads the programmed maximum velocity for trapezoidal mode.	
Examples:	Long form: **SOURce:PROFile:TRAPezoidal:MVEL?**	
	Short form: **MVEL?**	
Report:	6000.000[c/sec]	
CrossRef:	MC1401:	GET_VEL
	ShortCmd:	**SV?**

ACC <value>

Function:	This command sets the acceleration and deceleration for the trapezoidal mode.	
Range:	100.. 40.957.500 [c/s/s]	
Examples:	Long form:	**SOURce:PROFile:TRAPezoidal:ACC 80000**
	Short form:	**ACC 80000**
CrossRef:	MC1401:	SET_ACC
	ShortCmd:	**SA**

ACC?

Function:	Reads the programmed acceleration and deceleration for trapezoidal mode.
Examples:	Long form: **SOURce:PROFile:TRAPezoidal:ACC?** Short form: **ACC?**
Report (e.g.):	"50000.0095[c/s^2]"
CrossRef:	MC1401: GET_ACC ShortCmd: **SA?**

VEL:ACC <value>

Function:	This command sets the acceleration to be used with the velocity contouring mode.
Range:	0.. 40.957.500 [c/s]
Examples:	Long form: **SOURce:PROFile:VELocity:ACCeleration 77000** Short form: **VEL:ACC 77000**
CrossRef:	MC1401: SET_ACC ShortCmd: **SAV**

VEL:ACC?

Function:	Reads the programmed acceleration for the velocity contouring mode.
Examples:	Long form: **SOURce:PROFile:VELocity:ACCeleration?** Short form: **VEL:ACC?**
CrossRef:	MC1401: GET_ACC ShortCmd: **SAV?**

CURR:TPOS?

Function:	Reads the current target position generated by the profile generator. This query can be used in all profile generation modes: trapezoidal, S-curve, velocity and E-gearing.
Examples:	Long form: **SOURce:PROFile:CURRent:TPOSition?** Short form: **CURR:TPOS?**
CrossRef:	MC1401: GET_TRGT_POS ShortCmd: **GPP**

CURR:TVEL?

Function:	Reads the current target velocity generated by the profile generator. This query can be used in all profile generation modes: trapez, S-curve, velocity and E-gearing.
Examples:	Long form: **SOURce:PROFile:CURRent:TVELocity?** Short form: **CURR:TVEL?**
CrossRef:	MC1401: GET_TRGT_VEL ShortCmd: **GPV**

8.2.7 Filter Setting

PID <value_P>,<value_I>,<value_D>

Function:	This command sets the P-I-D filter values. Optional one, two or all three parameters can be set, depending on the position of the parameter value: "value_P,,value_D" if only P and D terms are to be defined, or ",,value_D" if only the D-term should be changed. all other filter terms remain unchanged. Each parameter string needs to have 2 commas to identify which value is which parameter. If the string in ot complete, an error message is queued.
Examples:	Long form: **SOURce:FILTer:GAIN:PID 220,30,2000** Short form: **PID 220,30,2000** Short form: **PID ,30,2000** Short form: **PID ,,2000** Short form: **PID 220,,2000** Short form: **PID 220,30,** Short form: **PID ,30,**
CrossRef:	MC1401: SET_KP, SET_KI, SET_KD ShortCmd: **DP**, **DI**, **DD**

PID?

Function:	Reads the set of P-I-D values of the digital filter.
Examples:	Long form: **SOURce:FILTer:GAIN:PID?** Short form: **PID?**
Report (e.g.):	"400,10,1000"
CrossRef:	MC1401: GET_KP, GET_KI, GET_KD ShortCmd: **DP?, DI?, DD?**

VFF <value>

Function:	This command sets the velocity feed forward gain of the digital filter.
Range:	0..32767
Default:	0
Examples:	Long form: **SOURce:FILTer:GAIN:VFF 210** Short form: **VFF 210**
CrossRef:	MC1401: SET_KVFF ShortCmd: **DF**

VFF?

Function:	Reads the velocity feed forward gain of the digital filter.
Examples:	Long form: **SOURce:FILTer:GAIN:VFF?** Short form: **VFF?**

Report (e.g.):	"120"	
CrossRef:	MC1401:	GET_KVFF
	ShortCmd:	**DF?**

BIAS <value>

Function: This command sets the bias of the digital filter. The BIAS value causes a constant torque offset to the motor. This can be helpful in vertical applications to compensate the static torque.

Examples: Long form: **SOURce:FILTer:BIAS 50**
Short form: **BIAS 50**

CrossRef:	MC1401:	SET_MTR_BIAS
	ShortCmd:	**DB**

BIAS?

Function: Reads the bias of the digital filter.

Examples: Long form: **SOURce:FILTer:BIAS?**
Short form: **BIAS?**

CrossRef:	MC1401:	GET_MTR_BIAS
	ShortCmd:	**DB?**

LIM:PERR <value>

Function: This command sets the largest position error allowed. If the limits are exceeded, the motion error handler is executed.

Examples: Long form: **SOURce:FILTer:LIM:PERR 4500**
Short form: **LIM:PERR 4500**

CrossRef:	MC1401:	SET_POS_ERR
	ShortCmd:	**SPE**

LIM:PERR?

Function: Reads the limit for position errors allowed.

Examples: Long form: **SOURce:FILTer:LIMit:PERRor?**
Short form: **LIM:PERR?**

CrossRef:	MC1401:	GET_POS_ERR
	ShortCmd:	**SPE?**

LIM:IERR <value>

Function: This command sets the integration limit for the digital filter.

Examples: Long form: **SOURce:FILTer:LIMit:IERRor 4500**
Short form: **LIM:IERR 4500**

Default: 2000

CrossRef:	MC1401:	SET_I_ERR
	ShortCmd:	**DL**

LIM:IERR?

Function:	Reads the integration limit of the digital filter.
Examples:	Long form: **SOURce:FILTer:LIMit:IERRor?** Short form: **LIM:IERR?**
Report (e.g.):	"2000 [c]"
CrossRef:	MC1401: GET_I_ERR ShortCmd: **DL?**

PERR?

Function:	Reads the current position error. This is the difference between the actual motor position and the position of the profile generator.
Examples:	Long form: **SOURce:FILTer:PERRor?** Short form: **PERR?**
CrossRef:	MC1401: GET_ACTL_POS_ERR ShortCmd: **TF**

IERR?

Function:	Reads the current integration error of the digital filter.
Examples:	Long form: **SOURce:FILTer:IERRor?** Short form: **IERR?**
Report (e.g.):	"5[c]"
CrossRef:	MC1401: GET_INTGR ShortCmd: **TI**

8.2.8 *I/O Commands*

OUTP <value>

Function:	This command allows to set the motor voltage without servo control. It writes directly to the D/A converter.
Examples:	Long form: **OUTPut 4500** Short form: **OUTP 4500**
Range:	-32767..+32767
CrossRef:	MC1401: SET_MTR_CMD ShortCmd: **SMO**

OUTP?

Function:	Reads the value written to the motor voltage output in all modes, so the servo output can be monitored dynamically.

Examples:	Long form: **OUTPUT?** Short form: **OUTP?**
CrossRef:	MC1401: GET_MTR_CMD ShortCmd: **SMO?**

OUTP:SERV {ON|OFF|0|1}

Function:	This command enables or disables the servo motor control. In the OFF state, the motor is not servo controlled and can be moves manually. Independent of the state, the encoder position is counted and can be read. When set to the OFF state, the BIAS value is set and the motor can be controlled by the OUTP command. When set to the ON state, the motor is servo controlled and the parameter set for the filter mode becomes effective. Alternatively to ON and OFF keywords, alse the numbers 0 (OFF) and not 0 (ON) can be used.
Examples:	Long form: **OUTPut:SERVo ON** Short form: **OUTP:SERV ON**
CrossRef:	MC1401: MTR_ON, MTR_OFF ShortCmd: **MN**, **MF**

OUTP:SERV?

Function:	Reads the current output status.
Examples:	Long form: **OUTPut:SERVo?** Short form: **OUTP:SERV?**
Report:	"ON", "OFF"
CrossRef:	MC1401: GET_STATUS (Bit 8) ShortCmd: **TS** (Bit 8)

OUTP:SIGN {DAC|PWM}

Function:	This command set the motor outputs to DAC16 or PWM.
Examples:	Long form: **OUTPUT:SIGNAL PWM** Short form: **OUTP:SIGN PWM**
CrossRef:	MC1401: SET_OUTP_PWM, SET_OUTP_DAC16 ShortCmd: **SOS{0,1}**

OUTP:SIGN?

Function:	Reads the current motor output mode.
Examples:	Long form: **OUTPUT:SIGNAL?** Short form: **OUTP:SIGN?**

CrossRef:	MC1401:	GET_OUTP_MODE
	ShortCmd:	**SOS?**

OUTP:LIM {<value>|MAX}

Function: This command defines an limit for the output voltage to the motor and can be used to limit the torque of the motor. Actually the motor *voltage* is limited to the programmed value.

It may be required also to lower the programmed velocity when setting the output limit. Since the motor voltage output is limited, the programmed velocity may not be reached. In this case the motion error increases and an unwanted motor stop due to exceeding motion error may occur.

The parameter MAX sets the output voltage range to the maximum value.

Range. 0 to 32767, corresponding 0 to 12V output voltage

Examples: Long form: **OUTPUT:LIMIT 6000**
Short form: **OUT:LIM 6000**

CrossRef:	MC1401:	SET_MTR_LIM
	ShortCmd:	**SQ**

OUTP:LIM?

Function: Reads the value for output limitation.

Examples: Long form: **OUTPUT:LIMIT?**
Short form: **OUTP:LIM?**

CrossRef:	MC1401:	GET_MTR_LIM
	ShortCmd:	**TQ**

8.2.9 *Macro Programming*

PROG?

Function: Reads the status of the Program mode. Report strings can be : "Running", Stopped", "xxxx loaded" or "No P. loaded"

Examples:	Long form:	**PROGRAM:STATE?**
	Short form:	**PROG?**

Report: "no program loaded"

PROG:DEF:BEG <progname>

Function: This command initializes the programming mode. During that mode is activated, all further commands transferred to the device are not executed but stored in the macro storage.

The programming mode is activated until the command "PROG:DEF:END" is send and the system returns to normal operating mode.

Stored macro commands can be called for execution by their name. The following list gives general hints to the programming mode:

Notes:

1. All commands transferred during the programming mode is activated are not executed but stored permanently in the memory.
2. Non valid commands or typographic error are recognized during input. These commands are not stored and an error message is placed in the error queue. These errors can be retrieved by "ERR?" after the programming sequence is terminated.
3. Logical errors generate a SYST:ERR during the runtime and place an error message into the error queue.
4. Up to 100 single commands can be stored in one macro command. Up to 16 named macro commands can be defined.
5. If the memory overflows during macro list input, the input sequence is terminated with the last command and an error is placed in the error queue.
6. If the C-844 is turned off before the programming mode is terminated, The macro sequence is not stored.
7. In order to store a macro under the same name like an already existing macro, the old one has to be deleted by "PROG:DEL <name>" before th new macro can be stored.

Examples:	Long form:	**PROGRAM:DEFINE:BEGIN mac1**
	Short form:	**PROG:DEF:BEG mac1**
CrossRef:	MC1401:	---
	ShortCmd:	---

PROG:DEF:END

Function: This command terminates a programming sequence. Afterwards, all further commands are executed at once.

Examples for a macro sequences see page 15.

PROG:DEL <name|ALL>

Function: This command clears the program storage with the indicated name. If no program was defined, an error message is generated.

Using the command with the parameter ALL, all stored macros will be erased.

Examples:	Long form**:**	**PROGRAM:DELETE myprog**
	Short form:	**PROG:DEL myprog**

CrossRef:	MC1401:	---
	ShortCmd:	---

PROG:CAT?

Function: Reads the names of all registered macro programs. The execution requires n communication cycles according to the number n of stored macro programs.

Examples:	Long form:	**PROGRAM:CATALOG?**
	Short form:	**PROG:CAT?**

PROG:LOAD <name>

Function: This command loads the named macro program for execution using the “PROG:RUN n” command, where the parameter n indicates the number of repetitions of the sequence.

If no program with the given name is found, an error is generated.

Examples:	Long form:	**PROGram:LOAD progname**
	Short form:	**PROG:LOAD progname**

PROG:RUN <n>

Function: Starts the execution of the indicated macro command for n times. It also restarts the macro if it was stopped by “PROG:STOP”.

During macro program execution, only Query commands are allowed. Exceptions are: STOP, HALT and PROG:STAT? commands

Examples:	Long form:	**PROGram:RUN 5**
	Short form:	**PROG:RUN 5**

PROG:STOP

Function: Stops a macro during execution.

Examples:	Long form:	**PROGram:STOP**
	Short form:	**PROG:STOP**

PROG:CONT

Function: Continues a stopped macro. If the macro is not loaded or nut running, an error messages is placed in the queue.

Examples:	Long form:	**PROGram:CONT**
	Short form:	**PROG:CONT**

PROG:LOAD:COMM?

Function: Reads the command sequence of the loaded program. For each command in the macro list one communication cycle is required.

Examples:	Long form:	**PROGram:LOAD:COMM?**
	Short form:	**PROG:LOAD:COMM?**

PROG:AUTO {<name>|NONE}

Function: This command starts the named macro sequence when power up.

When NONE as parameter is used, no macro is started automatically.

Examples:	Long form:	**PROGRAM:AUTO myname**
	Short form:	**PROG:AUTO myname**

PROG:AUTO?

Function: This command reports the name of the autostart program.

Examples:	Long form:	**PROGRAM:AUTO?**
	Short form:	**PROG:AUTO?**

8.2.10 *System Commands*

WAIT n

Function: This command halts the command execution for n milliseconds. It can be used to let the stage wait on a position for a defined time.

Examples:	Long form:	**SYSTEM:WAIT n**
	Short form:	**WAIT n**

"AXIS 1;TARG 50000;*WAI;WAIT 2000;TARG HOME"
This compound comand moves motor of axis #1 to position 50000 and let it rest there for 2 s before moving back home.

8.2.11 *Status Reporting*

PRES

Function:	Status preset. Makes all enable registers (OPER, MOT and QUES) transparent (sets all bits to 1).
Examples:	Long form: **STATUS:PRESET** Short form: **STAT:PRES**

MOT?

Function:	Reads the 16 bit motion event register. This register shows an event of the motion condition register. Any transition high-low and low-high are indicated.
Examples:	Long form: **STATUS:MOTION:EVENT?** Short form: **MOT?**
Report:	Reports an number in the range from 0 to 65536.

MOT?

Function:	Reads the 16 bit motion condition register. This register shows the motion states of all 4 axes: in motion, motion error, positive and negative limit switches.
Examples:	Long form: **STATUS:MOTION:CONDITION?** Short form: **MOT:COND?**
Report:	Reports an number in the range from 0 to 65536.

MOT:ENAB

Function:	Sets the enable mask for the motion event register to set the motion event summary bit in the status register.
Examples:	Long form: **STATUS:MOTION: ENABLE 16** Short form: **MOT:ENAB 16**

MOT:ENAB?

Function:	Reads the 16 bit motion event enable register.
Examples:	Long form: **STATUS:MOTION:ENABLE?** Short form: **MOT:ENAB?**
Report:	Reports an number in the range from 0 to 65536.

9 C-844 Register Structure

Registers are used to temporarily hold information regarding the operating and event status of the system. Host programs frequently use register information to coordinate motor movements and to handle exception errors. The C-844 status mechanism includes full implementation of the Event Status Register structure.

Besides the SCPI-defined Operation Status and Questionable Status registers and Motion status register is available with the associated condition, event and enable commands.

Register	Read	Write	Clear
IEEE 488.2 (8 bit) :			
Status Byte Register	*STB?	Firmware	*CLS
Service Request Enable Register	*SRE?	*SRE	*SRE 0
Standard Event Status Register	*ESR?	Firmware, no clear	*ESR?, *CLS
Standard Event Status Enable Register	*ESE?	*ESE	*ESE 0
SCPI (16 bit) :			
Operation Condition Register	OPER:COND?	Firmware	
Operation Event Register	OPER?	Firmware, no clear	OPER?, *CLS
Operation Event Enable Register	OPER:ENAB?	OPER:ENAB	OPER:ENAB, STAT:PRESet
Questionable Condition Register	QUES:COND?	Firmware	
Questionable Event Register	QUES?	Firmware, no clear	OPER?, *CLS
Questionable Event Enable Register	QUES:ENAB?	QUES:ENAB	QUES:ENAB, STAT:PRESet
C-844 defined Register (16 bit) :			
Motion Condition Register	MOT:COND?	Firmware	
Motion Event Register	MOT?	Firmware, no clear	MOT?, *CLS
Motion Event Enable Register	MOT:ENAB?	MOT:ENAB	MOT:ENAB, STAT:PRESet

Condition Registers

A condition register is a real-time, read-only register that constantly updates to reflect the current operating conditions of the C-844. For example, while if the positive limit switch of axis #3 is tripped, bit 10 of the motion condition register is set. When the limit switch is released, the bit clears.

Notes: - Reading the register let it unchanged.
- Bits are set and reset by firmware.
- Can be cleared by *CLS.

Event Registers

An event register is a latched, read-only register whose bits are set by the corresponding condition register. Once a bit in an event register is set, it remains set (latched) until the register is cleared by a specific clearing operation. The bits of an event register are logically ANDed with the bits of the corresponding enable register and applied to an OR gate. The output of the OR gate is applied to the status byte register.

Notes: - Event registers can not be set by command.
- Event registers are cleared when they are read.
- All event registers are cleared at power-on or by the *CLS command.

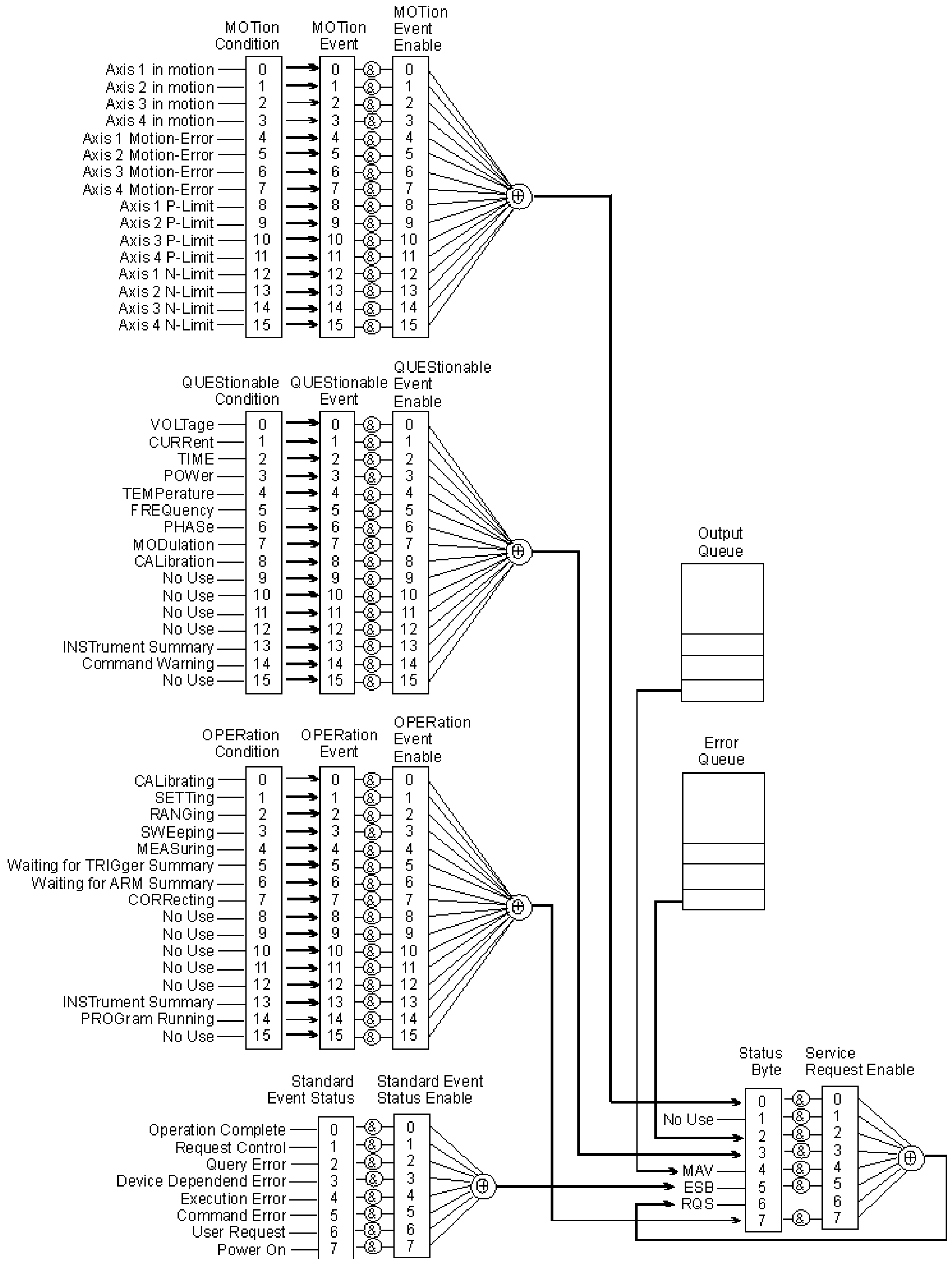
Enable registers

An enable register is programmed by the user ans serves as a mask for the corresponding event register. An event bit is masked when the corresponding bit in the enable register is cleared, When masked, a set bit in an event register cannot set a bit in the *Status Byte Register*.

To use the Status Byte Register to detect events, you must unmask the events by setting the appropriate bits of the enable registers.

Notes: - Enable Registers are not cleared when read.
- Can be read and write by commands.
- Can be cleared by command with parameter NULL
- Can be cleared by SYSTem:PRESet

C-844 Registers: IEEE 488.2 and SCPI Status Registers



DWG.: C844STAT.WMF

10 Appendix

10.1 Pin Assignments

Connector: "RS-232", "AUX" :

internal J3, J4	C-844 rear panel	Function
type: PCB mounted	**type: DSub9(m)**	
	1	DCD (in), not used
	6	DSR (in), not used
	2	**RxD (in)**
	7	**RTS (out)**
	3	**TxD (out)**
	8	**CTS (in)**
	4	DTR (out) (*)
	9	RI (in), not used
	5	**GND**

(*) this +5V output can used for external purposes.

Connectors: "Motor 1", "Motor 2", "Motor 3", "Motor 4" :

internal: J8,J9,J10,J11	C-844 rear panel	Function
type: IDC16	**type: DSub15(m)**	
PIN 1	1	out: +12V, (motor brake)
2	9	Motor (-)
3	2	Motor (+)
4	10	Power-GND
5	3	MAGN
6	11	SIGN
7	4	out: VCC +5 V
8	12	negative Limit
9	5	positive Limit
10	13	REFS
11	6	Limit GND
12	14	Encoder: A(+) / ENCA
13	7	Encoder: A(-)
14	15	Encoder: B (+) / ENCB
15	8	Encoder: B (-)
16		n.c.

Connector: "Joystick/Analog In" :

internal J7 type: IDC16	C-844 rear panel type: DSub15(m)	Function
PIN 1	1	Vcc
2	9	Vcc
3	2	SW1A
4	10	SW1B
5	3	XA
6	11	XB
7	4	GND
8	12	GND
9	5	GND
10	13	YB
11	6	YA
12	14	SW2B
13	7	SW2A
14	15	Vcc
15	8	Vcc
16		n.c

Connector: "Digital I/O"

internal J6 type: IDC26	C-844 rear panel type: DSub25(m)	Function
PIN 1	1	GND
2	14	Input channel 0
3	2	Input channel 1
4	15	Input channel 2
5	3	GND
6	16	Input channel 3
7	4	Input channel 4
8	17	Input channel 5
9	5	GND
10	18	Input channel 6
11	6	Input channel 7
12	19	DIN Clock
13	7	GND
14	20	Output channel 0
15	8	Output channel 1
16	21	Output channel 2
17	9	GND
18	22	Output channel 3
19	10	Output channel 4
20	23	Output channel 5
21	11	GND
22	24	Output channel 6
23	12	Output channel 7
24	25	DOUT clock
25	13	GND
26		GND