Operating Manual MS 33E

C-808

DC-Motor Controller
Operating Manual MS 33E

C-808

DC-Motor Controller

This Document is valid for this Product:

C-808.00   DC-Motor Controller
C-808.50   DC-Motor Controller

Release:   4.41
Release Date:  22 August 1996
This Manual describes the C-808.xx DC Motor Controllers with the Layout Version "F", delivered since February 1994.

Table of Contents:

1. Introduction .................................................. 4
   1.1. C-808 Features: ........................................ 4
   1.2. C-808 Models and Accessories ......................... 5

2. DC-Motors ..................................................... 5
   2.1. Resolution ................................................ 6
   2.2. Motor / Encoder Wiring .................................. 6

3. Starting Operation ........................................... 7
   3.1. Operating the C-808 ..................................... 7
   3.2. Troubleshooting ......................................... 7

4. Input / Output Capabilities .................................. 8
   4.1. Digital I/O Channels .................................... 8
       4.1.1. Programming I/O Channels ......................... 8
       4.1.2. Output TTL Signals: ............................... 9
   4.2. Analog Input ............................................ 9
   4.3. Limit switches .......................................... 9
       4.3.1. Connecting Limit Switches ......................... 10
   4.4. Joystick ................................................ 10
   4.5. Hand-Held ASCII Terminal .............................. 11
   4.6. Online LCD Position Display ............................ 12

5. Communication ............................................... 13
   5.1. RS-232 Communication ................................ 13
       5.1.1. Set up RS-232 Communication : .................... 15
       5.1.2. Programming Example for RS-233 Communication : 15
       5.1.3. RS-232 Handshake Lines : .......................... 16

6. How the C-808 Works ......................................... 18
   6.1. System Performance .................................... 18
   6.2. Servo control .......................................... 18
   6.3. Speed Control ......................................... 18
   6.4. Default Values ........................................ 19
7. C-808 Hardware .............................................................................. 20
  7.1. Firmware Version .................................................................. 20
  7.2. Motor Current Output Stage .................................................. 20
  7.3. DIP Switch Settings ............................................................... 20
  7.4. C-808 Power Requirements .................................................. 21
  7.5. Component Diagram ............................................................ 22

8. Software Tools ............................................................................ 23
  8.1. PITERM: Terminal Emulation Program ................................ 23
  8.2. MoveMaster 4.5 Operating Program ................................... 23

9. Commands .................................................................................. 23
  9.1. Single Commands .................................................................. 24
  9.2. Compound Commands ......................................................... 25
  9.3. Macro Commands ............................................................... 25
  9.4. Single Character Commands ............................................... 26
  9.5. Reporting Commands .......................................................... 27

10. Starting Communications ......................................................... 27

11. Calculating Acceleration Ramps ............................................... 28

12. COMMAND SET REFERENCE .................................................. 30

13. Connectors on C-808.00 and C-808.50 Controllers ................... 50
1. **Introduction**

C-808 series DC-Motor Controllers are positioning and motion control systems for small DC-motors. Two motors up to 25 Watt can be driven without external amplifiers. Furthermore the C-808 controls up to 8 I/O lines and 4 analog inputs for general purposes. These features characterize the C-808 as a versatile control system for laboratory use or general testing purposes in development, production and quality control.

Additional accessories like hand-helt terminals, joysticks and a wide range of DC-motors are available.

More than 80 commands are implemented in the firmware to control movements and related parameters. All internal registers can be reported by commands. Commands may also be compounded and/or defined as macros for their easy recall and execution.

RS-232 communications port is provided for control from a terminal or host computer. Using the built-in interpreter, commands can be transferred in ASCII characters. This allows you the use of a simply structured but powerfully capable command set.

1.1. **C-808 Features:**

- Simultaneous 2-axis control
- On-board amplifiers for motors up to 3 Amps and up to 36 Volts
- Relative and Absolute positioning
- Speed control
- Large RAM space for macro commands
- Learn mode
- 4 A/D converters (8-bit) on-board
- 8 programmable I/O channels
- Limit switch and reference inputs
- Multi-drop RS-232 interface
- Encoder supply
- DIP switch setting for default values

Position feedback is accomplished by incremental encoders connected either to the motor shaft or directly to the stage. All standard types may be used as long as they output two channels in quadrature (90 degrees shifted). These outputs may be either sine waves or TTL signals.

PI's main catalog lists various motor/encoder combinations from .25 to 25 Watts.
1.2. **C-808 Models and Accessories**

C-808.00  DC-Motor Controller standard version
C-808.50  DC-Motor Controller with 96 pin main connector, OEM version to be use with a prepared motherboard. This version has a shorter pcb. Connectors J3, J4, J6, J7, J8 and Dip switch S1 are not installed. All connections run through the main connector.

Accessories:

C-808.92  Update Set for upcoming Firmware releases
C-808.80  LCD Display for C-808
C-808.31  RS-232 Cable, 1.5 m, 10pin/DB9
C-816.00  Pushbutton keyboard box
C-817.00  Hand-Held RS-232 Terminal (ASCII)
C-819.00  Analog Joystick
C-819.80  Adapter Cable for analog joysticks

Software:

C-800.91  MoveMaster Operating Program

2. **DC-Motors**

The C-808 drives DC-motors up to 30 Watts average power by its internal amplifiers.

TTL encoder signals are used to control the motor position in a feedback servo loop. Encoder signals must be related to the motor position (angle). These signals can be generated either from a rotary encoder attached to the motor shaft or from a linear encoder connected to the moving stage. To enable the controller to stabilize the motor position, no backlash can be tolerated between encoder reading and motor axis.

In most applications, motors with shaft-connected encoders are used. The C-808 will work with sine wave and square wave signals as well, if the encoder emits two phase-shifted signals in the way most encoders work (shifted by 90 degrees). The encoder signals must be TTL-level; that is, they must swing below 1.5 volts and above 3.5 volts to be recognized. This hysteresis band is required for noise immunity by the encoder interface.

Most of PI's positioning stages can be equipped with DC-motor drives. Different motor/encoder combinations can be used. We recommend using the following motor/encoder combinations:
For each type of encoder a specific set of some parameters is required. In detail, the gain (SG and SE), the velocity (SV) and the acceleration (SA and SD) have to be set according the encoder line number to avoid unstable conditions.

### 2.1. Resolution

The step resolution of the C-808 depends on the line number of the used motor/encoder combination. Each line generates two slopes on both encoder channels A and B so that in total 4 edges are counted for one line. The step resolution equals 4 times the line number of the encoder. Encoder with 100 lines/rev generate 400 step counts, with 500 lines 2000 step counts/rev.

### 2.2. Motor / Encoder Wiring

Motors used with PI stages have different cable connectors depending on the special stage. Standard DC-motors with encoders have a 10 pin IDC connector:

**MOTOR CABLE PIN ASSIGNMENTS**

<table>
<thead>
<tr>
<th>PIN</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Motor (+)</td>
</tr>
<tr>
<td>2</td>
<td>+5 Volt</td>
</tr>
<tr>
<td>3</td>
<td>Encoder channel A</td>
</tr>
<tr>
<td>4</td>
<td>Encoder channel B</td>
</tr>
<tr>
<td>5</td>
<td>Encoder GND</td>
</tr>
<tr>
<td>6</td>
<td>Motor (-)</td>
</tr>
<tr>
<td>7</td>
<td>Limit switch right</td>
</tr>
<tr>
<td>8</td>
<td>Limit switch left</td>
</tr>
<tr>
<td>9</td>
<td>Reference signal</td>
</tr>
<tr>
<td>10</td>
<td>Logic GND</td>
</tr>
</tbody>
</table>

DC-Motors with more than 6 Watt have separate DC current lines to be connected to the wire clamps on the C-808 board.
3. Starting Operation

3.1. Operating the C-808

C-808 DC-Motor Controllers is ready to use and can be installed within minutes. All you need is a power supply, cable for communication and a data terminal or host computer to operate the RS-232 interface.

Follow these steps to run the C-808 DC Motor Controller:

1. Connect the motor/encoder combination. For small motors the flat ribbon cable C-815.4x /C-814.6x can be used. Larger DC-motors require separated lines for the motor current. Note that the motor cables have line 1 (pin 1) marked with a black line. Plugging the motor cable in the wrong way may cause damage to the encoders.

2. Verify the DIP switch settings. See chapter 7.3

3. Connect the RS-232 communications cable (PI order # C-815.31). See chapter 5.1 for RS-232 cable pin assignment.

4. Connect the power supply and turn it on. For power requirements see chapter 7.4: Specifications.

After power up the connected motors may get a short voltage pulse causing a movement of some degrees. But then they must stay at rest. Whether the motors are in the MN or MF state (position controlled or not) depends on the DIP switch setting.

5. Run PITERM terminal emulation programs, (supplied on PI-TOOLS disk). Type in the first command "VE" and the C-808 should respond with the version message. At this point, the C-808 should be ready to execute commands to be verified working, and you may go to PART B - PROGRAMMING GUIDE to learn how to communicate with the controller.

3.2. Troubleshooting

If you encounter problems in operating the C-808 Controller, go through the following check list. Most obstacles are caused by minor operating errors. Better understanding of internal functions give you a chance to overcome bothering obstacles.

The following list may give you some helpful hints to avoid and how to fix problems:

PROBLEMS & SOLUTIONS

1. Motors are running at top speed and are not controllable.

Reset the board using the reset point on the C-808. A possible reason is that encoders count in the wrong direction. Use the TE command several times to find out if position information is available from the encoders. If no change occurs in the error report, check the connection from the encoder to the board. If the error is increasing instead of decreasing, exchange the plus (+) and minus (-) lines on the motor connector.

2. Motors are oscillating at the end of movement.
Set static gain to a lower value (e.g. SG 20)

95% of these problems are related to bad cables, or wrong connections with the computer. Make sure that the computer is IBM compatible and use special cable for either XTs and ATs.

4. Set velocity (SV) command does not effect velocity of motor.
Verify that the motor/encoder combination is able to run the programmed speed (refer to section 2.2, motors). If yes, check the Following Error (TF) several times while the motor is running. If the Following Error is decreasing, the motor runs at full speed and tries to catch up with the virtual position. When the Following Error reaches zero, the motor will run at the programmed velocity. If the Following Error is increasing, the programmed velocity is too large for that motor. Even at full voltage the virtual position increases its distance from the current position. Set velocity to a very small value such as 100 and watch the Following Error. It should now decrease.

4. Input / Output Capabilities

4.1. Digital I/O Channels
The C-808 Controller has 8 digital I/O lines which can be programmed as input or output. These lines are accessible via the J7 connector on the C-808 board. (See pin table in PART C - APPENDIX). These I/O lines are capable of sinking a current of 1.6 mA. They will not be damaged when shorted to GND.

4.1.1. Programming I/O Channels
Program the I/O lines with the following commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF</td>
<td>Channel OFF</td>
</tr>
<tr>
<td>CI</td>
<td>Channel IN</td>
</tr>
<tr>
<td>CN</td>
<td>Channel ON</td>
</tr>
<tr>
<td>CT</td>
<td>Channel OUT</td>
</tr>
<tr>
<td>WC</td>
<td>Wait channel</td>
</tr>
<tr>
<td>XF</td>
<td>Execute IF channel is OFF</td>
</tr>
<tr>
<td>XN</td>
<td>Execute IF channel is ON</td>
</tr>
</tbody>
</table>

For a complete description of these and other program commands please refer to PART B - PROGRAMMING GUIDE.

As default, after power-on, all I/O lines are defined as inputs and are pulled to +5 Volt. This status can be read by the TC (Tell Channel) command, which causes an output of an 8 bit value:

Command: TC (rtn)  
Report: 255 CRLF ETX (ETX)
If channels 1 and 6 are pulled to GROUND, the report command will tell you the values 254 and 223, respectively. If you have selected HM (Hex Mode) for your input and output, the same test would yield the hex values "FE" and "DF."

Commands:

TC1 read 0 or 1 [ 0==low, number==high]
TC2 read 0 or 2
TC3 read 0 or 4
... TC8 read 0 or 128

4.1.2. Output TTL Signals:
All lines can be defined as outputs with the CT (Channel OUT) command. To define only line n as output, give the command CTn. All lines defined as output remain at 5 Volts. With CNn (Channel ON==high) or CFn (Channel OFF==low), the output of these lines can be set to high or low. If you don't specify a channel, the command sets all channels.

Example: To cause channel 5 to output 5 Volt, send:
CT5,CN5 (rtn)

4.2. Analog Input
C-808 Controllers have 2 analog inputs with 8 bit resolution. The TA (Tell Analog) command reads the value of all 4 input lines and the command TAn reads the value of input line n.

The voltage applied at the input line is compared with a reference voltage at the reference input (+5V). The result is converted to a numerical value (0 to 255) which can be read with the TA command. This value represents the ratio of the input to the reference voltage.

The analog inputs of the C-808 are located on J5 and J6.

4.3. Limit switches
C-808 Controllers support right and left limit switches for each axis. They provide a software-based limit switch implementation, where the contact of a limit switch is latched in software and will not be released until a move command is transferred. This leads to reliable recognition of the switches even if they bounce.

When a limit switch line is pulled to GROUND, the controller stops this motor immediately and reduces the target by a programmable amount from the position at which contact was made, which causes the motor to step back. The original target will be erased. The number of steps to be stepped back can be programmed with the LS command. The default value is 4,000 steps.
Every motor-driven stage which has a mechanical stop should be equipped with two limit switches—one on either side. The relation of right and left side limit switches to the actual mechanical side depends on the gear head, which sometimes reverses the direction. You should test the behavior of limit switch safety operation before working without operator control.

4.3.1. Connecting Limit Switches

Each of the 10-pin motor connectors has two limit switch lines for that axis (see pin assignments in PART C -- APPENDIX). To separate the limit switch lines from the 10-line motor cable, we offer a special split cable (PI part number C-815.60) which splits the 10 lines in 4 and 6 for direct connection to the limit switch socket on a mechanical stage.

Limit Switch Connection

![Limit Switch Connection Diagram]

Pin 10  Pin 8  Pin 7

4.4. Joystick

Another optional method for driving the motors is a joystick. This option offers a very convenient way to handle XY tables. The voltage applied to the motor is related to the analog output value of the joystick. While one or both motors are driven manually via the joysticks, the current position of all axes can be read by the TP command.

Each motor can be driven by an analog voltage input from 0 to +5 Volt. At a median position of 2.5 Volts, plus/minus a small deadband, no driving voltage is applied to the motor.

The C-819 Joystick is a special type of IBM compatible analog joysticks, modified to be referenced to GND (standard IBM compatible joysticks do not work as voltage divider rather than as variable resistors).
The Joystick can be connected to connector J6 on the C-808 using the C-819.80 adapter cable.

To operate the motor via the joystick, the motor status has to be "MN" (Motor On). Two different joystick gain tables are available to match the C-819.00 joystick as well as IBM compatibles. Transfer the following sequence to enable joystick operation for axis n:

For C-819.00 joysticks:
- nMN (rtn) sets motor in Motor_On state
- nJN1 (rtn) selects linear joystick table

For standard IBM compatible joysticks:
- nMN (rtn) sets motor in Motor_On state
- nJN (rtn) selects nonlinear joystick table

To disable joystick operation, transfer JF (Joystick OFF).

To move the motor, press pushbutton for the selected channel and operate the stick. Motor movement only occurs while the button is pressed. Then the target position is moved and the motor follows the target under velocity control.

ORDERING INFORMATION:
C-819.00 Analog Joystick for two axes to be used with C-808

4.5. Hand-Held ASCII Terminal
For applications that do not otherwise require a PC, the C-817 Hand-Held ASCII Terminal can be used. All commands can be typed in and reports are shown on a liquid crystal alphanumeric display with 4 rows of 20 characters.

C-817 Specifications:
- Display: 4 x 20 character LCD display
- Power Supply: +5 Volts supplied by external supply.
- Cable: Coiled cable, length 6 feet.
- Connector: 25-pin D-Sub connector.
Using C-817 Terminal:

When the Terminal is first connected, go through the following sequence:

- Connect Terminal and switch power ON.
- Press CTRL+SHIFT+F1. A menu appears on the display.
- Press F2.
- Press F2 again. The display shows PARITY=EVEN.
- Press F1 three times to change to PARITY = SPACE.
- Press F2 three times. If the display reads ECHO = ENABLED, press F1 to disable echo.
- Press F2. If the display reads HANDSHAKE: ENABLED, press F1 to disable handshake.
- Press F5 to save the parameters.

Now the settings are:

BAUD= 9600; DATA Bits = 8; PARITY = SPACE.

Ordinarly, the keys you press are sent to the C-808 without being displayed on the terminal. If you want the commands you enter to show on the display, press EN <ENTER> to turn the C-808 internal ECHO mode ON.

4.6. Online LCD Position Display

Beginning with the EPROM firmware version 4.0, dated July 22th, the C-808 controllers have the option of supporting 4x16 character LCD displays for online position reading. The display can be connected to the I/O lines connector J7 on the C-808 board. The display is available from Physik Instrumente as a standard product (order# C-808.80), ready for use with the appropriate cable.

Note, that the LCD Display is using the I/O lines and cannot be used for other purposes. If you need the I/O lines for your application, the LCD display has to be disabled by setting the DIP switch #3 to OFF.

Hints for using the LCD display:
1. Verify EPROM version C808V40 (Version 4.0 or later) and PAL U5C (in position U5).
2. Set DIP switch #3 to ON
3. Verify that trim potentiometer R5 is installed (5k Ohm)
4. Plug in the display cable to J7.
5. Power on the C-808 and the online display is activated.

In case you're working with an older firmware version, you should order an update set for the C-808 with the order # C-808.92.

5. Communication

5.1. RS-232 Communication

The firmware version 4.2 supports full handshaking on character and command level as well as an 256 Bytes input buffer.

Although RS-232 Communication is very easy to handle presumed that cables and port parameters are setup properly, there could arise some problems in cable wiring and programming. This note may be valuable to you for understanding the way how C-800 series controllers handle RS-232 data transfer and how to program command level handshaking.

The hardware handshake guarantees every character sent to the device to be accepted and processed. This does not necessarily mean that it makes sense to send a character, because this character may be interpreted in a specific way depending on the busy status of the controller.

The user has to be aware that a terminated command line takes some time for internal processing. During that period of time, the hardware handshake allows to send to the controller because you may want to interrupt the running macro command. Exactly at this point, the user should be aware that the CTS line of the controller indicates the ready status (I'm ready to accept a character) although the controller is busy because of executing a command line. If any character is sent to the controller, it will be interpreted as a termination character and the processing of the command string will be suspended and the whole string will be canceled. Be aware that this character works only as a command terminator and must not be part of a new command.

Example: "1MR45000,1WS80,TP,1GH,1WS80,TP"

To perform the motion of 45000 counts the controller needs around 5 seconds. Assuming 3 seconds after sending the first command line (pressing the (rtn) key) the next command is transferred: "TG" (rtn).

This command would cause the following: Because the first command is still running, the first character of the second command, the "T", is interpreted as a termination character that terminates and erases the first command line. The active command (1MR45000) will be fully executed, but now the rest of the command line is empty. Not empty at all, because the second character of the second command, the "G" is there. But the "G" is neither a valid command nor the beginning of a valid command, and that means, the controller is responding with an error message, sending the "?" to the PC.
How to avoid this? First you could modify the second command by inserting a space character as the first character: " TG". Now the space (ASCII 32) would work like the termination character and the trailing "TG" can be processed in the right way. But this doing makes something the user may not want to do: It breaks and erases the first command. More likely he would have intended to complete the first command line to its full extend and then the second command should be executed.

How to do this?

Although the RTS/CTS handshake is working fine, this handshake does not handle the command level handshaking. On the one hand, a running complex compound command should be terminated every time the operator want it to, on the other hand a new command should be send as soon as possible to the controller without interrupting the previous command.

To allow the right timing, the controller sends an ETX (or any other character defined by the CM command) after the command processing is completed. The user only has to wait for this termination character and the next command string may be transferred. This handshaking is called 'Command Level Handshaking'.

As default, an ETX is send when a compound command is completed and to indicate that the command interpreter is ready to take the next command. Also an ETX is sent to terminate a report. The last characters of a report are CR LF ETX (13 10 03). If the input command has had no repeat command and no wait command for further actions, an additional ETX is sent the ready status for the next command as explained above.

Examples:

<table>
<thead>
<tr>
<th>Command</th>
<th>Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;1TP&quot; (rtn)</td>
<td>&quot;01Pxxxxxxxxxx&quot; CR LF ETX ETX</td>
</tr>
<tr>
<td>&quot;TP&quot; (rtn)</td>
<td>&quot;01Pxxxxxxxxxx&quot; CR LF</td>
</tr>
<tr>
<td></td>
<td>&quot;02Pxxxxxxxxxx&quot; CR LF ETX ETX</td>
</tr>
</tbody>
</table>

The last ETX character can and should be used to time the command sequences to the controller while the handshaking of the command string itself using the RTS/CTS lines may be optional and does have some importance for programming languages like C which do not offer input buffering on the PC side.

Because some users may like it better to have the second ETX (used for command timing) to be replaced by another character. For thoses we have implemented the command "CMn" where n represents any ASCII character up to 31.
Example:
"CM2" (rtn) : sets the handshake character to STX, ASCII 2.

Now the report looks like this:

<table>
<thead>
<tr>
<th>Command</th>
<th>Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;1TP&quot; (rtn)</td>
<td>&quot;01Pxxxxxxxxxx&quot; CR LF ETX STX</td>
</tr>
<tr>
<td>&quot;TP&quot; (rtn)</td>
<td>&quot;01Pxxxxxxxxxx&quot; CR LF</td>
</tr>
<tr>
<td></td>
<td>&quot;02Pxxxxxxxxxx&quot; CR LF ETX STX</td>
</tr>
</tbody>
</table>

Also the bell character '7' can be used for that purpose. Mind you this character is not sent to the screen but gives a beep sound when it's received.

5.1.1. Set up RS-232 Communication:
From factory, the RS-232 interface is set to the following parameters:
9600 baud, 8 data bits, 1 stop bit, no parity

In QuickBASIC one of the following open commands can be used:

<table>
<thead>
<tr>
<th>No Handshake</th>
<th>Open Command</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OPEN &quot;COM1:9600,N,8,1,CS,DS,CD&quot; FOR RANDOM AS #1</td>
</tr>
</tbody>
</table>

Using Handshake: OPEN "COM1:9600,N,8,1,CS100,DS,CD" FOR RANDOM AS #1

5.1.2. Programming Example for RS-233 Communication:

```
' Program "RSDEMO.BAS"
' Compiler: Microsoft QuickBasic 4.5
' Version : 1.0, 20.01.94
' Physik Instrumente (PI) GmbH, D-76337 Waldbronn, FAX ++49 7243-604 145

' Example program to demonstrate how to program the C-800 Series Controllers
' with RS 232 communication. All firmware versions released since
' December 1993 will work.

CLS
PRINT "Start of Program RSDEMO.BAS"
OPEN "COM1:9600,N,8,1,CS100,DS,CD" FOR RANDOM AS #1

PRINT #1, "CM2" : CHR$(13); 'This command defines the
GOSUB HandleHandshake
' command execution termination character.
' As default an ETX (ASCII 3) is sent and
' this is changed
' to the STX (ASCII 2) character.
' Both lines only need to be sent once time
' while the device is powered up.

' Definition of commands:

cmdline1s = "lmr200000,ls80,1tp,lg3,ls80,1tp"
```

Release 4.4                         Physik Instrumente (PI)
5.1.3. RS-232 Handshake Lines:

If you want to use the handshake signals, make use of the two handshake lines as described below:

The C-808 supports full hardware handshake for sending and receiving data. This can be wired as an RTS/CTS handshake. A standard Null-Modem cable with crossed RTS/CTS, RxD/TxD lines can be used.

There are two pins on the J3 connector that are used for handshaking:
PIN 5: **Handshake in**: (To be connected with RTS on PC side). Permission for C-808 to send; PC is ready to take a data byte.

PIN 7: **Handshake out**: (To be connected with CTS on PC side). C-808 is ready to take a data byte

**RS-232 Cables for C-808**:

The C-808 controller has a 10 pin IDC socket (connector J3) for RS-232 communication (for Pin assignment see Appendix C). Use the PI-standard C-815.31 RS-232 cable to connect the PC.

**Pin Assignment C-815.31 RS-232 Cable**:

<table>
<thead>
<tr>
<th>PC-side</th>
<th>C-808 side</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB 9pin (F)</td>
<td>10 pin IDC</td>
</tr>
<tr>
<td>Pin 1 (DCD)</td>
<td></td>
</tr>
<tr>
<td>Pin 2 (RxD)</td>
<td>Pin 3 (TxD)</td>
</tr>
<tr>
<td>Pin 3 (TxD)</td>
<td>Pin 1 (RxD)</td>
</tr>
<tr>
<td>Pin 4 (DTR)</td>
<td></td>
</tr>
<tr>
<td>Pin 5 (DSR)</td>
<td></td>
</tr>
<tr>
<td>Pin 5 (GND)</td>
<td>Pin 9 (GND)</td>
</tr>
<tr>
<td>Pin 7 (RTS, out)</td>
<td>Pin 5 (Handshake in)</td>
</tr>
<tr>
<td>Pin 8 (CTS, in)</td>
<td>Pin 7 (Handshake out)</td>
</tr>
<tr>
<td>Pin 9 (nn)</td>
<td></td>
</tr>
</tbody>
</table>

**RS-232 Cable (DB25(m) to C-808)**:

<table>
<thead>
<tr>
<th>PC-side</th>
<th>C-808 side</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB 25pin (M)</td>
<td>10 pin IDC</td>
</tr>
<tr>
<td>Pin 1</td>
<td></td>
</tr>
<tr>
<td>Pin 2 (TxD)</td>
<td>Pin 1 (RxD)</td>
</tr>
<tr>
<td>Pin 3 (RxD)</td>
<td>Pin 3 (TxD)</td>
</tr>
<tr>
<td>Pin 4 (RTS, out)</td>
<td>Pin 5 (Handshake in)</td>
</tr>
<tr>
<td>Pin 5 (CTS, in)</td>
<td>Pin 7 (Handshake out)</td>
</tr>
<tr>
<td>Pin 6 (DSR)</td>
<td></td>
</tr>
<tr>
<td>Pin 7 (GND)</td>
<td>Pin 9 (GND)</td>
</tr>
<tr>
<td>Pin 8 (DCD)</td>
<td></td>
</tr>
</tbody>
</table>

**Ordering Information for PI-Standard Cable**:

**C-815.31** RS-232 Flat Ribbon Cable, 1.5 m for C-808 to AT connection with full handshake.
6. How the C-808 Works

6.1. System Performance

The C-808 Controller is able to simultaneously control position and velocity of up to two DC-motors. Each axis is related internally to a target position, actual position, error and velocity register, each 32 bits wide. The position register is related to the encoder counter and always contains the current motor position, whether or not the motor is in the MN (Motor ON) state.

The target register is updated by commands like MR (Move Relative) or MA (Move Absolute). The error register contains the difference between the target and the position register.

Other registers contain values for gain, torque, dynamic gain, and dynamic error. Motor controlling is done by analog output rather than the more common Pulse Width Modulation, for reduced audible and electrical noise. The control voltage generated by 16-bit D/A converters is amplified to drive the motors either in the CW or CCW direction (push-pull operation). D/A converters generate an analog voltage proportional to the value in the error register.

The voltage on the motors depends on the values for error, gain and velocity. As the difference of target and position grows larger, the motor driving voltage also becomes larger. The value for the gain corresponds to the slope of the amplification.

The C-808 Controller always controls the position of the motor when the MN mode is active. During a move, the velocity is controlled by calculating the ideal position at which the motor should be at all times to follow the desired velocity profile. Changes in the external torque cause a change in the driving voltage to maintain the velocity at a constant value. The motor driving voltage can be set to a value which can be limited by the SQ (Set TORQUE) command.

6.2. Servo control

Servo control of position is accomplished by a proportional control algorithm. Encoder signals are read every servo loop cycle from the internal 12-bit registers of the HCTL2000 counters and merged into the 32-bit position registers in the RAM. The encoder reading is done throughout all operations and is valid as long as power is applied to the C-808.

The contents of the target register, which is controlled by the processor, only changes by executing a command like MR, MA or DH. The difference between the position and target registers is calculated and stored in the error register.

Within an error range from -4096 to +4096 (maximum) the operating voltage on the motor is proportional to that value. If the error exceeds the value set by the SQ command, this value is output to the motor. The slope of the proportional amplification depends on the Set Gain (SG).

The gain parameter, which can be modified with the SG command, defines the stiffness of the motor position coupling—the ratio of error counts to drive voltage. Within the proportional range, the motor driving voltage depends on the error counts and the gain value SG.

The maximum proportional range is 13-bits wide (from -12 to +12-bits). The actual proportional range depends on the programmed torque and gain parameters. At the largest torque value of 127, and the smallest gain of 1, the proportional range covers -4096 to +4096 error counts. The default gain value is 32 and gives good motor performance for low inertial loads such as gear heads.

6.3. Speed Control

The velocity of each motor may be programmed separately by issuing the command SV (Set Velocity) for each axis. The programmed value gives the velocity in steps per second.

When the motor is started with a Move command, the controller tries to keep the motor movement on the position/time line. The slope of that line depends on the programmed speed. If the motor
movement is slowed by external forces for a short time and then released, the motor speeds up to catch the position/time relation. So, the speed control is actually a position in time control.

The difference between the virtual (calculated) position and the actual position is stored in the dynamic error register. It can be read with the TF (Tell Following error) command. Increasing dynamic error indicates that the mechanical load on the motor exceeds its capabilities. Decreasing TF values indicate that the motor catches up to the virtual position. It is possible for the velocity to exceed the programmed value while it is catching up. The SQ command can be used to limit the maximum expected velocity.

Speed control can only work properly if the motor voltage can be increased sufficiently to compensate for increased external loads. Therefore the programmed velocity (SV) should usually not be programmed in excess of 80% of the maximum free-running speed of the motor at the maximum output voltage, depending on the magnitude of expected load changes.

To check the free-running speed of the motor, set the velocity to a value which cannot be reached, e.g. SV500000, then move the motor and ask for the speed with TV. This free-running speed depends on the motor, the load, and the line count of the encoder.

### Free Running Motor Speeds:

<table>
<thead>
<tr>
<th>Model #</th>
<th>Lines/Counts/</th>
<th>Speed [counts/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rev. Rev.</td>
<td></td>
</tr>
<tr>
<td>C-120.40</td>
<td>15 60</td>
<td>8000</td>
</tr>
<tr>
<td>C-120.80</td>
<td>15 60</td>
<td>8000</td>
</tr>
<tr>
<td>C-124.40</td>
<td>15 60</td>
<td>8000</td>
</tr>
<tr>
<td>C-124.50</td>
<td>500 2000</td>
<td>200000</td>
</tr>
<tr>
<td>C-124.51</td>
<td>100 400</td>
<td>38000</td>
</tr>
<tr>
<td>C-132.50</td>
<td>500 2000</td>
<td>170000</td>
</tr>
<tr>
<td>C-132.51</td>
<td>100 400</td>
<td>35000</td>
</tr>
</tbody>
</table>

### 6.4. Default Values

When powered up, the C-808 controller sets all working registers to the default values. Some values are read from the DIP switch settings.

In detail, the controller comes up in the following status:

- All channels in the Motor OFF (MF) state.
- Echo is OFF (EF).
- Gain is set to 32 (TG,SG).
- Dynamic gain is set to 32 (TN,SE).
- Torque is set to 127 (TQ,SQ). This is the maximum value.
- Acceleration is set to 200,000 (TL,SA)
- Deceleration is set to 200,000 (TD,SD)
- Velocity is set to 100,000 (TY,SV)
- Deceleration point is set to 25,912 (TR,SP)
7. C-808 Hardware

7.1. Firmware Version

The C-808 Controller is based on an extended EPROM firmware, which supports more than 80 commands for motor control and I/O handling.

The firmware is subject to improvement and customer needs. If you have comments or suggestions about the firmware, please advise us. Our representatives will gladly forward your remarks to our development department.

7.2. Motor Current Output Stage

Each DC-motor is driven by a Burr-Brown power operational amplifier supplied with a voltage of +/-12 to +/-40 volts (We recommend to not operate the C-808 with more than 18 V without to adapt some other components to this voltage. In this case, consult the factory). The outputs are short-circuit protected as well as offering current limiting and over-temperature protection. The current limit is normally set to 2.5 to 3.5 A at the factory, but can be changed on special order.

7.3. DIP Switch Settings

DIP Switch S1 (8 bit): (also accessible on 96-pin connector)

<table>
<thead>
<tr>
<th>Switch</th>
<th>Function</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1-1 (22c)</td>
<td>Limit switch enable</td>
<td>(ON==enable)</td>
</tr>
<tr>
<td>S1-2 (21a)</td>
<td>Joystick &amp; Pushbutton-keyboard enable</td>
<td>(ON==enable both)</td>
</tr>
<tr>
<td>S1-3 (21b)</td>
<td>LCD display enable</td>
<td>(ON==enable)</td>
</tr>
<tr>
<td>S1-4 (21c)</td>
<td>Non volatile RAM installed</td>
<td>(ON==installed)</td>
</tr>
<tr>
<td>S1-5 (20a)</td>
<td>I Term enable</td>
<td>(ON==enable)</td>
</tr>
<tr>
<td>S1-6 (20b)</td>
<td>Baud Rate</td>
<td></td>
</tr>
<tr>
<td>S1-7 (20c)</td>
<td>Baud Rate</td>
<td></td>
</tr>
</tbody>
</table>
C-808 DC Motor Controller Operating Manual MS 33E

S1-8 (19a) Baud Rate

Switches S1-6 to S1-8 select the Baud Rate:

<table>
<thead>
<tr>
<th>S1-6</th>
<th>S1-7</th>
<th>S1-8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>19200</td>
<td>9600</td>
<td>4800</td>
</tr>
<tr>
<td>2400</td>
<td>1200</td>
<td>600</td>
</tr>
<tr>
<td>300</td>
<td>9600</td>
<td></td>
</tr>
</tbody>
</table>

If no Dip switch is connected to the board like in case of the C-808.50 version, the baud rate is selected to 9600 baud. This is the situation when bits 6/7/8 are all OFF.

Dip Switch S2 (4 bit)

Bit-Switch #:

<table>
<thead>
<tr>
<th>S2-1</th>
<th>S2-2</th>
<th>S2-3</th>
<th>S2-4</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>X</td>
<td>Network Address 128</td>
</tr>
<tr>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>X</td>
<td>Network Address 129</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>X</td>
<td>Network Address 130</td>
</tr>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>X</td>
<td>Network Address 131</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>OFF</td>
<td>X</td>
<td>No Network, Motors are in MF state after power on reset.</td>
</tr>
</tbody>
</table>

X: any position

7.4. C-808 Power Requirements

Power Requirements:

J8 pin1: +12 V...+18 V 100 mA + total motor current
J8 pin2: -12 V -18 V 100 mA
J8 pin3: GND 500 mA
J8 pin4: +5 V
7.5. Component Diagram
8. Software Tools

Programming the C-808 DC-Motor Controller means to transfer commands to the device in order to start a motion, to modify a parameter, to download a macro file or to ask for a report.

Communications may be accomplished via the RS-232 interface which is standard in all C-800 series controllers.

The easiest way to establish communications is to use one of the communications programs supplied with the controller. It is not necessary to develop communications programs for manual control applications. Only for special program implementation a certain amount of programming is required.

8.1. PITERM: Terminal Emulation Program

PITERM (current version is 5.4) is a terminal emulating program supporting RS-232 communication for all PI devices (DC-Motor controllers and PZT control interfaces) using serial communications. Optional handshake can be used if required.

Each character typed in is sent to the device and each character received from the device is displayed on the screen. The program works in full duplex mode using COM1, COM2 or COM3 port in your PC.

PITERM is available on the PITOOLS distribution disk that comes with all PI's DC-Motor Controllers.

8.2. MoveMaster 4.5 Operating Program

MoveMaster is a MS-DOS program, designed to work with the C-800 series controllers. It controls all current parameters, displays position and motor speed on-line and allows for modification of motor motion by sending commands or downloading macros.

MoveMaster allows you to work either in workscreen or terminal mode.

Using the workscreen mode, a table containing all relevant parameters is displayed on your EGA or VGA monitor. For both DC-motor axes, the table contains current position, programmed speed (PSPEED), actual speed (ASPEED), programmed dynamic and static gain (DGAIN, GAIN), deceleration ramp (DECELTN) and the current motor status in binary format.

In the command line, you may edit a compound command by using the editing keys and send it to the controller with the ENTER key. Messages from the controller, such as reports or error notes, are displayed in the report line. Reports concerning valid values for the mask will be displayed immediately in the table.

ORDERING INFORMATION:

C-800.91 MoveMaster Operating Program

9. Commands

More than 80 commands are available for programming the C-808 DC-motor controller. Use these commands for motion and position control, reporting, voltage output, and voltage input.

Each command is composed of an axis number [optional], a two-character function abbreviation, and often a parameter number. A carriage return CR (rtn) signals the end-of-command to the controller.

Commands may be executed in various ways:

- Single command -- One function
Terms Used

Throughout this manual certain abbreviations and standards will be used:

- **CMD** is the two-character command mnemonic. Actual commands are indicated in bold type such as: TP
- **a** is the axis number.
- **[ ]** brackets indicate that the value is optional. If no axis number is specified, the command will affect both axes.
- **n** is a parameter value.
- **ETX** is an ASCII end-of-text character (03) sent by the C-808 to indicate the completion of a Tell command data transmission.
- **CRLF** carriage return and line feed characters (13 & 10) sent by the C-808 to indicate the start of a new line.

Commands are shown without quotation marks to avoid confusion regarding whether or not they are part of the command. Indicated responses from the C-808 are enclosed in quotes for clarity.

9.1. Single Commands

A single command consists of an axis number [optional], a two-character function, a parameter number (if required), and a terminating (rtn) carriage return. This type of command is executed immediately after the carriage return is received. Using a single command type, the motor movement may be controlled by transmitting command after command.

The syntax for a single command is:

```
[a] CMD [n] (rtn)
```

Examples:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1MR2000</td>
<td>Move motor#1 2,000 steps relative to the present target</td>
</tr>
<tr>
<td>2MN</td>
<td>Set motor 2 in ON state</td>
</tr>
<tr>
<td>2GH</td>
<td>Send motor 2 to home position (go home)</td>
</tr>
<tr>
<td>1TP</td>
<td>Report (tell) position for motor 1</td>
</tr>
<tr>
<td>1MA20000</td>
<td>Send motor 1 to absolute position 20,000</td>
</tr>
<tr>
<td>2tp</td>
<td>Report (tell) the position of motor 2</td>
</tr>
<tr>
<td>tp</td>
<td>Report (tell) the position of both motors</td>
</tr>
</tbody>
</table>

Both uppercase and lowercase characters are valid, and spaces are allowed.
The leading axis number is optional for all commands except the motion commands Move Relative (MR) and Move Absolute (MA). These two "move" commands are excluded from this capability for safety reasons.

- **MR2500 (rtn)**: Incomplete command for Move Relative. Must have an axis number.
- **1MR2500 (rtn)**: Correct command includes axis number.
- **1TP(rtn)**: Reports "01P000000000" CRLF ETX ETX
- **tp (rtn)**: Reports:
  - "01P000000000" CRLF
  - "02P000000000" CRLF ETX ETX

**9.2. Compound Commands**

A compound command is a series of single commands separated by commas rather than by a carriage return. In this way, it is possible to string together several commands before terminating with the carriage return. These multiple commands will be executed sequentially.

The syntax for a compound command is:

\[ [a]CMD[n], [a]CMD[n], ..., ..., (rtn) \]

Examples:

- **1MR2000,2MR4300 (rtn)**
- **1m5000,wa500,2ma12000,wa800,tp (rtn)**
- **2 ma 3500,2 ws 77,wa 120,tp (rtn)**

A compound command such as the following example may be entered as one program line. It instructs the motor attached to axis 1 to move 1,000 encoder counts in the positive direction, wait in that position 500 milliseconds, return to the original position, wait 1 second, and then repeat the cycle 5 times.

"1MR1000,WS77,WA500,1MR-1000,WS77,WA1000,RP5" (rtn)

Once this command is entered, it remains in the buffer until replaced by another command and can be re-executed by transferring a carriage return. Compound commands may contain up to 35 single commands.

When an axis number is given in any command, that number is assumed for the remainder of the compound command. This is also true for macro commands. When no axis number is specified, the command interpreter assumes axis number 0, or both axes are referenced.

Example: "1MR1000, 2MR2000,WS77,TP" (rtn)

Many first time users will issue a command of this type expecting the position of both axes to be reported after both have stopped for 77 milliseconds. In reality, this command is stored internally as 1MR1000,2MR2000,2WS77,2TP. If stored as a macro command, this will be obvious when the macro is displayed.

If the intent was to wait for and report on both axes, the command should be entered as: "1MR1000,2MR2000,0WS77,TP" (rtn).

**9.3. Macro Commands**

Macros can be a most powerful tool for the programmer. A macro command is a grouping of commands to form a short program, implemented by a macro name.
To use macros for programming the C-808 controller, insert an MD (Macro Definition) command as the first instruction in the command string. The syntax for macro commands is:

MD(macro#), followed by a compound command string:
MD [n], ... , ... ( rtn )

Example:
MD3,1MR1000,WS77,WA500,1MR-1000,WS77,WA1000,RP5 (rtn)

In this example, MD3 defines macro #3. To call up this macro, just issue the command EM3 or MC3. Execute Macro #3 or Macro Command 3 respectively. EM and MC are interchangeable. We include both forms based on customer preferences. (Other interchangeable commands are MN/GO and MF/ST.)

Macro commands may be stored in any order, but you may prefer to number them sequentially as they are entered, because the system gives no warning if you define (and overwrite) an existing macro. You may wish to do this under many conditions, such as when one macro is called by another. It is sometimes desirable to define a complex motion in one macro and define key parameters such as torque, gain, or velocity, in another macro which is called by the main macro.

Macro commands can call another macro, but only one level deep. For instance, MC1 could call MC2, but MC2 could not then call MC3. A macro may call as many other macro commands as desired, as long as each one called does not call another.

Example: MD1,MC2,MC3,MC4,MC5,MC6

Macro commands may contain up to 34 single commands. In the RAM a total number of 4096 Bytes are reserved to store up to 128 Macros. This is enough storage for an average macro command length of four single commands. Each single command requires 7 bytes of storage space, with one additional byte per macro to store the length. Therefore, longer macros are more efficient, but not as many can be stored. The 4096 bytes allocated for macro storage overlaps with the storage allocated for strings. If less than the maximum number of strings is stored, more macros can be used, and vice versa.

9.4. Single Character Commands

Ordinarily the controller may be interrupted during the execution of a compound command simply by entering any character. However, there are times when it is desirable to access information pertaining to the status of an operation without interrupting the operation.

For that reason, a special set of commands is available which do not interfere with the current command execution. These single character commands cause the controller to report the following values for both axes:

$ == TA (tell analog)
% == TS (tell status)
& == TV (tell velocity)
# == TC (tell channel)
@ == TP (tell position)
+ == TE (tell error)
* == AB (abort)
9.5. Reporting Commands

Reporting commands cause the C-808 controller to emit a string of data, whether a position, target, help or other information. These commands are easy to remember as they usually begin with a Tell statement. For example, TT (Tell Target), or TP (Tell Position). For a detailed list of all reporting commands, see Section 13.3.

All Tell commands query a certain register on the board, which may be 8, 16, 24 or 32-bits long. Accordingly, the reports have different formats. The following examples assume that DM (Decimal Mode) has been selected. In the first example, if HM (Hex Mode) had been selected, there would be only 8 digits after the letter 'T.'

Example for 32 and 24-bit register query:
Transmit: 1TT (rtn)
Receive: "01T0000000000" CRLF ETX ETX

Transmit: TT (rtn)
Receive: "01T0000000000" CRLF
"02T0000000000" CRLF ETX ETX

Example for 8 and 16-bit register query:
Transmit: 1TG (rtn)
Receive: "01G0032" CRLF ETX ETX

Transmit: TG (rtn)
Receive: "01G0032" CRLF
"02G0032" CRLF ETX ETX

10. Starting Communications

At power-on the C-808 Controller is ready to receive data. A VE (Version) command is preloaded into the command buffer so that if the first character sent to the board is a carriage return, the board responds with the version number message. (For VE description see Section 13.3)

The first command to test the communications should be a TT command. Type in a TT command. On your terminal monitor the following characters should be displayed:

"01T0000000000"
"02T0000000000" CR LF ETX ETX

The first two digits 01 and 02 give the axis number. The T says that the following number contains the target data. After reporting both requested targets an ETX (End-of-Text) character, equivalent to CHR$(3), is emitted. And last an ETX is send to indicate that the C-808 is ready for the next command.
Every report is terminated by an ETX character. When using the PiTERM Terminal Program, the values for the different axes will be displayed in the same row, as CRLF (carriage return line feeds) are stripped off.

When you get the proper response (as above) on your monitor, the board is ready to work. Now all registers are loaded with their default values. Most tasks may be accomplished using only these default values—other tasks may require value modifications.

Now check the servo loop motor operation. Connect one or both motors according to the motor wiring instructions in PART A. The motor should be at rest and it should be movable by hand to any position (the motor position servo loop is not yet activated).

11. **Calculating Acceleration Ramps**

For many applications, the rate of acceleration and deceleration is not critical, so the built-in values are adequate and need not be changed. Systems that use small motors—such as the Micro-Mo DC Micrometer Drive, with a 2 Watt motor—have very low inertia, and thus, no need for long acceleration and deceleration ramps to prevent oscillation. However, larger motors with high inertia loads will require setting. For this reason the C-808 offers several commands related to speed control:

- **SA** = Set acceleration
- **SD** = Set deceleration
- **SP** = Set point of deceleration
- **CP** = Calculate point to begin deceleration
- **TL** = Tell acceleration
- **TD** = Tell deceleration
- **TR** = Tell deceleration point

**Example for use of the ramp-functions:**

![Graph showing ramp functions](image)

The distance that must be travelled is 130,000 steps. During that movement the maximum speed is 20,000 steps/second, and must be reached after 40,000 steps. To set the value for acceleration work through the following steps:
From the formulas for constant acceleration:

\[ s = \frac{a}{2} \cdot t^2 \quad \text{and} \quad v = a \cdot t \]

we get \[ a = \frac{v^2}{2s} \quad s = \frac{v^2}{2a} \quad \text{and} \quad v = \sqrt{2sa} \]

distance: \( s \) (steps)
velocity: \( v \) (steps/sec)
acceleration: \( a \) (steps/sec/sec)

\[ => a = \frac{v^2}{2s} = \frac{(20,000 \ \text{steps/sec})^2}{2 \cdot 40,000 \ \text{steps}} \]
\[ => a = 5,000 \ \text{steps/sec}^2 \ \text{(command SA 5000)} \]

For deceleration we need a value of 10,000 steps/sec (SD10,000).
The starting point of the deceleration is calculated in this way:
\[ s = \frac{v^2}{2a} = \frac{(20,000 \ \text{steps/sec})^2}{2 \cdot (10,000 \ \text{steps/sec}^2)} = 20,000 \ \text{steps} \]

The deceleration point must be set at a distance of 20,000 steps using the command SP 20000.
12. COMMAND SET REFERENCE

(All commands available in firmware version C808 V4.3 or later)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>Abort motion: Stop the motor &amp; set target = Position</td>
</tr>
<tr>
<td>AP</td>
<td>Adjust Position: Modifies position in point storage</td>
</tr>
<tr>
<td>BR</td>
<td>Baud Rate: Set baud rate</td>
</tr>
<tr>
<td>CF</td>
<td>Channel OFF: Set output channel to 0 Volt</td>
</tr>
<tr>
<td>CI</td>
<td>Channel IN: Define channel as input</td>
</tr>
<tr>
<td>CM</td>
<td>Change Terminator: Changes command completion character</td>
</tr>
<tr>
<td>CN</td>
<td>Channel ON: Set output channel to 5 Volt</td>
</tr>
<tr>
<td>CP</td>
<td>Calculate Point: Calculates the deceleration ramp</td>
</tr>
<tr>
<td>CS</td>
<td>Check sum: Reports the EPROM checksum</td>
</tr>
<tr>
<td>CT</td>
<td>Channel OUT: Define channel as output</td>
</tr>
<tr>
<td>DB</td>
<td>Dead Band: Set dead band</td>
</tr>
<tr>
<td>DH</td>
<td>Define Home: Define zero position</td>
</tr>
<tr>
<td>DM</td>
<td>Decimal Mode: In / Out in decimal mode</td>
</tr>
<tr>
<td>DS</td>
<td>Define String: Defines a string for later use</td>
</tr>
<tr>
<td>EF</td>
<td>Echo OFF: Set echo off</td>
</tr>
<tr>
<td>EM</td>
<td>Execute Macro: Starts macro execution</td>
</tr>
<tr>
<td>EN</td>
<td>Echo ON: Set echo on</td>
</tr>
<tr>
<td>FE</td>
<td>Find edge: Move to reference point</td>
</tr>
<tr>
<td>GH</td>
<td>Go home: Move to the origin</td>
</tr>
<tr>
<td>GO</td>
<td>Go: Start the motor</td>
</tr>
<tr>
<td>HE</td>
<td>Help: Reports all available commands</td>
</tr>
<tr>
<td>HE1</td>
<td>Help: Reports all motion commands</td>
</tr>
<tr>
<td>HE2</td>
<td>Help: Reports all parameter setup commands</td>
</tr>
<tr>
<td>HE3</td>
<td>Help: Reports all reporting commands</td>
</tr>
<tr>
<td>HE4</td>
<td>Help: Reports macro and sequence commands</td>
</tr>
<tr>
<td>HE5</td>
<td>Help: Reports miscellaneous commands</td>
</tr>
<tr>
<td>HM</td>
<td>Hex Mode: In / Out in hexadecimal mode</td>
</tr>
<tr>
<td>IF</td>
<td>I-Term off: Turns I-Term off, leaves P-Term unchanged</td>
</tr>
<tr>
<td>IN</td>
<td>I-Term on: Enables integral term for exact positioning</td>
</tr>
<tr>
<td>IO</td>
<td>Increment Output: Increment output voltage</td>
</tr>
<tr>
<td>JF</td>
<td>Joystick OFF: Disables joystick operation</td>
</tr>
<tr>
<td>JN</td>
<td>Joystick ON: Enables joystick operation</td>
</tr>
<tr>
<td>LF</td>
<td>Limit OFF: Disables limit switch operation</td>
</tr>
<tr>
<td>LI</td>
<td>Learn Position: Learn position and increment number</td>
</tr>
<tr>
<td>LN</td>
<td>Limit ON: Enables limit switch operation</td>
</tr>
<tr>
<td>LP</td>
<td>Learn Position: Stores actual position</td>
</tr>
<tr>
<td>LS</td>
<td>Limit Set: Set count number to step back</td>
</tr>
<tr>
<td>LT</td>
<td>Learn Target: Stores actual target</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>MA</td>
<td>Move absolute</td>
</tr>
<tr>
<td>MC</td>
<td>Macro Command</td>
</tr>
<tr>
<td>MD</td>
<td>Macro Definition</td>
</tr>
<tr>
<td>MF</td>
<td>Motor OFF</td>
</tr>
<tr>
<td>MI</td>
<td>Move to point</td>
</tr>
<tr>
<td>MN</td>
<td>Motor ON</td>
</tr>
<tr>
<td>MP</td>
<td>Move to point</td>
</tr>
<tr>
<td>MR</td>
<td>Move relative</td>
</tr>
<tr>
<td>PS</td>
<td>Print String</td>
</tr>
<tr>
<td>RG</td>
<td>Reset string</td>
</tr>
<tr>
<td>RM</td>
<td>Reset Macro</td>
</tr>
<tr>
<td>RP</td>
<td>Repeat</td>
</tr>
<tr>
<td>RS</td>
<td>Reset points</td>
</tr>
<tr>
<td>RT</td>
<td>Reset</td>
</tr>
<tr>
<td>SA</td>
<td>Set Acceleration</td>
</tr>
<tr>
<td>SD</td>
<td>Set Deceleration</td>
</tr>
<tr>
<td>SE</td>
<td>Set dynamic gain</td>
</tr>
<tr>
<td>SF</td>
<td>Slave OFF</td>
</tr>
<tr>
<td>SG</td>
<td>Set Gain</td>
</tr>
<tr>
<td>SN</td>
<td>Slave ON</td>
</tr>
<tr>
<td>SO</td>
<td>Set Output</td>
</tr>
<tr>
<td>SP</td>
<td>Set Point</td>
</tr>
<tr>
<td>SQ</td>
<td>Set torque</td>
</tr>
<tr>
<td>ST</td>
<td>Stop</td>
</tr>
<tr>
<td>SV</td>
<td>Set Velocity</td>
</tr>
<tr>
<td>TA</td>
<td>Tell Analog</td>
</tr>
<tr>
<td>TB</td>
<td>Tell Board</td>
</tr>
<tr>
<td>TC</td>
<td>Tell Channel</td>
</tr>
<tr>
<td>TD</td>
<td>Tell Deceleration</td>
</tr>
<tr>
<td>TE</td>
<td>Tell Error</td>
</tr>
<tr>
<td>TF</td>
<td>Tell Following error</td>
</tr>
<tr>
<td>TG</td>
<td>Tell Gain</td>
</tr>
<tr>
<td>TI</td>
<td>Tell Iterations</td>
</tr>
<tr>
<td>TL</td>
<td>Tell Acceleration</td>
</tr>
<tr>
<td>TM</td>
<td>Tell Macro</td>
</tr>
<tr>
<td>TN</td>
<td>Tell Dynamic Gain</td>
</tr>
<tr>
<td>TO</td>
<td>Tell Output</td>
</tr>
<tr>
<td>TP</td>
<td>Tell Position</td>
</tr>
<tr>
<td>TQ</td>
<td>Tell Torque</td>
</tr>
<tr>
<td>TR</td>
<td>Tell Decel point</td>
</tr>
<tr>
<td>TS</td>
<td>Tell Status</td>
</tr>
<tr>
<td>TT</td>
<td>Tell Target</td>
</tr>
<tr>
<td>TV</td>
<td>Tell Velocity</td>
</tr>
<tr>
<td>TY</td>
<td>Tell Velocity</td>
</tr>
</tbody>
</table>

Release 4.4  Physik Instrumente (PI)  Page 31
TZ Tell Macro Zero
VE Tell Version
WA Wait Absolute Delay
WC Wait Channel
WD Wait Delta
WE Wait for Error
WF Wait Channel OFF
WN Wait Channel ON
WP Wait for Position
WS Wait for Stop
XF Execute if ch OFF
XN Execute if ch ON

Command Set in Alphabetic order

'a' represents the axis number (1 or 2) and may be omitted except for the MR and MA commands.

'n' represents an integer parameter to be sent with the command.

aAB Abort Motion
This command performs an emergency stop. The motor stops immediately and deletes any further motion requirements for that axis. The target position for the selected axis is changed to be equal to the present position. Abort Motion is used for stopping an undesired motion as well as to retain a position.

Example: AB (rtn) : Aborts motion of both motors
2AB (rtn) : Aborts motion of motor 2

AP Adjust Position
Adjust Position is used to modify a previously stored coordinate point to a modified new position. This command will update the currently active position (i.e. -- the last position moved to).

Example: Assume, Point #10 was X1,Y1
MP10 (rtn)
1MR10,2MR-20 (rtn)
AP (rtn)

The new location of point #10 will be (X1+10), (Y1-20) and is now stored as position 10.

BRn Baud Rate
When the C-808 board is powered on, the baud rate for serial communications is read from the DIP switch settings. This command allows you to change the baud
rate while the system is running. The actual baud rate is determined by using the value in a countdown circuit.

This command takes effect immediately, so use it with caution. Any value entered will result in some baud rate change but not necessarily a standard one.

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>19,200</td>
<td>2</td>
</tr>
<tr>
<td>9600</td>
<td>5</td>
</tr>
<tr>
<td>4800</td>
<td>11</td>
</tr>
<tr>
<td>2400</td>
<td>23</td>
</tr>
<tr>
<td>1200</td>
<td>47</td>
</tr>
<tr>
<td>600</td>
<td>95</td>
</tr>
<tr>
<td>300</td>
<td>191</td>
</tr>
</tbody>
</table>

CFn Channel n OFF
Causes channel n to go to logic zero level (0 Volt). Before using this command, channel n must be defined as output with the CT (Channel OUT) command.
Example: CT5, CF5 (rtn): Defines channel 5 as output and sets it to 0 Volt level.

CIn Channel n Input
Defines channel n as an input channel. This command must be issued before using the command TC (Tell Channel).
Example: CI7 (rtn)

CMn Change Terminator
This command allows the user to select any character up to ASCII 32 to be send as a command terminator. How to use it, see section 'RS-232 communication'.

CNn Channel n ON
Causes channel n to go to logic-one level (5 Volts). Before using this command, channel n must be defined as an output channel with the CT command (Example: CN7 (rtn)).

aCP Calculate Point
This command calculates the point where the deceleration ramp has to start to achieve a continuous and smooth deceleration to the target position, depending on the programmed values for velocity and deceleration. The Calculate Point command helps you to avoid overshooting the motor and makes for an easy, smooth approach to the final position.

The CP command should be issued after you have defined the deceleration (SD) and set velocity (SV).

To report the calculated values, use the TR command. The reported values refer to long moves where the motor reaches its full velocity. For short moves where the motor does not reach the programmed speed, the calculated value will be overridden by a number that equals half the move.
Therefore, the CP command will work for long and short moves. For short moves there is only one restriction—the acceleration (SA) value should be equal to the deceleration (SD) value.

Sending the CP command is only required after the deceleration or velocity has been changed.

Example: sa60000, sd60000, sv34000 (rtn)
          cp, tr (rtn)

Report: "01R00000009984" CRLF
         "02R00000009984" CRLF ETX

A long move would be a move of more than twice the calculated point value, in this case more than 19968 counts.

CS Checksum

Reports a value that corresponds to the contents of the ROM. It is used as a quick self-test. The value reported varies with the release version of the firmware and verifies the release number. Another command to show the firmware release is the VE (version) command.

CTn Channel n OUT

Similar to CI (Channel In) command. Used to define channel n as an output. Must be used before the CN (Channel ON) or CF (Channel OFF) commands are used for that channel.

Example: CT8 (rtn)

aDBn Dead Band (0 < n < 127)

If the motor is within +n or -n counts from the target position the motor current is shut off. In other words, if the position error is less + or - n counts, the motor is in the MF state. It may be moved by hand.

The deadband command only increases the deadband limits. To set to lower limits, the SG (Set Gain) or RT (Reset) command must be issued first.

Example: 1DB100 (rtn)

aDH Define Home

Defines the current motor position as Zero. The zero position (home position) is n counts off the current position. If no value for n is given, the home position is set to the current motor position.

Examples:

DH (rtn): Sets the current motor position for both axes to 0.
2DH5000 (rtn): Sets the current position for axis 2 to 5000.

When a value is given for n, the results may be unexpected. The lower 12 binary bits of the position count are held in hardware counters which cannot be preset to any value other than zero, greatly reducing the value of this feature.
DM Decimal Mode
The C-808 Controller can handle both decimal and hexadecimal numbers. The DM command sets the controller in the decimal mode and all inputs are interpreted as decimal numbers and the system reports requested values also in decimal.
The decimal mode is active after powering up.
Example: DM, SG100 (rtn)

DSn Define String
Up to 128 strings may be defined by the DS command. For string storage a total of 3,840 bytes are reserved in the RAM.
Example: DS1 (rtn) Defines String #1
"THIS IS A MESSAGE STRING" (rtn)
With that command the string "THIS IS A MESSAGE STRING" is stored to string #1. Now it may be displayed whenever the PS (Print String) command is issued.
Example: 2MA25000, 2WS77, PS1, WA5000, 2MA100000 (rtn)
Please remember that this command is unique in the C-800 Family in that it requires a carriage return character to be entered after the command, and before the data. Failure to enter the (rtn) after the DSn can lead to some confusing situations.
After the DSn (rtn) is entered, any sequence of characters may be entered and will be stored as string #n. Storage continues until either a (rtn) character or a (CTRL C) is entered. In the latter case, the string will be displayed without a trailing CRLF sequence when it is called by a subsequent PSn command. This is quite useful when it is desired to display multiple messages on the same line.
Example: DS16 (rtn)
The position of axis one is (CTRL C)
PS16, 1TP (rtn)
The latter command will print
"The position of axis one is 01P0000000000 CRLF ETX"

EF Echo OFF
Causes suppression of all non-essential data output. Only data specifically requested from the C-808 will be transmitted. On power-up, the C-808 controller is set with to EF mode (active).

EN Echo ON
Causes all characters received through any active communications port to be echoed to that port as received. Normally used when operating with the host in full duplex mode.

EMn Execute Macro n (also MCn)
Either of these commands may be used to implement a previously defined macro command. They are identical in operation, except that MC may be used in a compound command with other commands in the instruction set. If there is no macro defined by the number n, no action will be taken.
Example: EM3 (rtn) : Executes macro #3
Before executing EM3, that macro must be defined (MD).

**aFEn Find Edge**

This command is used to initialize the system at a given position. The motor runs at a programmed speed until a transition occurs on the reference input line. The direction of motion depends on the value of n.

Example:
- 2FE1 ( rtn ) : Causes axis 2 to move in a positive direction.
- 2FE0 ( rtn ) : Causes axis 2 to move in a negative direction.

This command assumes that your system has the reference line connected. (See PART C - APPENDIX for connections.)

**aGH Go Home**

The Go Home command causes the specified axis to move to absolute zero position. Equivalent to an aMAO (Move Absolute to zero) command with the exception that both axes may be commanded simultaneously.

Example:
- GH ( rtn ) : Moves both axes to zero
- 1MA0 ( rtn ) : Moves axis 1 to zero

**aGO Motor Go**

Functionally equivalent to the MN (Motor ON) command. Causes one or both axes to begin or resume servo control toward the assigned target position. If the target position is different from the actual position, the motor will attempt to move.

Examples:
- 1GO ( rtn )
- GO ( rtn )

**HEn Help**

If n equals zero, this command reports the valid mnemonics for the installed firmware version. It lists all two-letter commands supported by your firmware version.

The Help function may be called with any number from 1 to 5. While the HE0 command lists only the short form abbreviations of the commands, the HE[n] command gives you more detailed information.

HE1 reports all motion commands
HE2 reports all parameter setup commands
HE3 reports all reporting commands
HE4 reports all macro and sequence commands
HE5 reports miscellaneous commands

**HM Hexadecimal Mode**

After transmitting this command, all inputs and outputs are in hexadecimal mode.

Example: HM,SV7FFF ( rtn )
Set hex mode and transfer velocity values in hex number.

**aIN**  
I-Term ON

**aIF**  
I-Term OFF

Using mechanical stages where the transmission system shows some friction, the exact positioning to zero step error can't be obtained just by using the proportional term. An error count number depending on the frictional force will be reported using the TE command. In most cases the integral term (l-term) brings the solution. The l-term causes the motor voltage to increase at the end of a move until the exact target is reached. At zero error counts the extra current is shut off and the stage remains the desired position without annoying error counts.

Try to find out whether the enabled l-term is useful for your application.

**aOn**  
Increment Output by the value of n

This command adds the value of n to the present output voltage setting. The value of n may range from zero to full scale.

If it is desired to produce a square wave at the maximum output voltage of the C-808, a command to set the output at maximum (1S032768), followed by (1I01,1O-1,RP), will swing the output back and forth across the boundary.

**aJN**  
**aJF**  
Joystick ON/OFF

Joystick operation uses the four analog inputs for motor movement. If the controller is in the JN (Joystick ON) status, the four analog input readings are used as control signals for motor driving commands.

Example: MF,JN (rtn) : Enable all motors for joystick operation.

Before switching to joystick operation, the designated motor must be set in the MF (Motor OFF) status.

1JF (rtn) : Disable joystick operation for axis 1.

Now the analog inputs may be used again for general purposes.

Please be aware that issuing a JN command with no joystick connected will cause the motors to run at full speed and they will not be stopped by the limit switches!

**Lln**  
Learn Position Incrementing

This command allows a series of sequential positions to be stored as a string starting at n and having a number of points as defined by the repeat command. Ll is especially useful for storing a series of traced points that you want to be repeated. Ll may be implemented with motor on or motor off. Both axes are stored.

Example: L11,WA100,RP99 (rtn)

This command line will store a position every 100 milliseconds up to position 100. A subsequent command MP1,WA100,RP99 will move to the 100 defined positions in sequence, from 1 to 100.
LPn  **Learn Position**

The LP command is used for storing the coordinates of the current position of both axes for later use by the MP and MI commands.

aLSn  **Limit Set**

Sets the value of the increment to be subtracted from the position at which the system encounters the limit switch for axis a. In other words, if limit switches are in use, the normal operation is for the system to back away from the first point at which the limit switch made contact. If no LS is set, the default value is 4,000 steps.

For example, axis a hits a limit switch while moving in the positive direction at a position of 12,000. The target position is immediately changed to be 4,000 counts less than the position at which the switch made contact. The new target position is now 8000 (12,000-4000). Since the axis is beyond this new target, it will reverse the direction and back away from the switch. A value of 4,000 is usually enough to ensure that the limit switch re-opens. If not, the LS command may be used to change the back-step number.

Example: 2LS8000 (rtn)

1LS12000,2LS20000 (rtn)

When moving in the negative direction, the increment is added instead of being subtracted from the point at which the contact was made, so that operation is always to reverse direction whenever a limit switch is contacted.

The limit switch implementation is a "soft" implementation, designed to provide an automatic reversal of motion when encountered. This is done so that the operation under computer control cannot damage the mechanical components by inadvertently commanding them past the useful end of travel.

Keep in mind, that this implementation does not provide absolute safety. Under normal conditions, it reliably protects the equipment, however, it may not control motion under some types of hardware failure. To be entirely safe, limit switches must be used which interrupt the flow of current to the motor. They do not require connection to the C-808 controller.

Limit switch operation can be enabled by use of the LN (Limit ON) command at any time.

LTn  **Learn Target**

Similar to the LP command, LT does not require actually moving to the position to store the point. Learn Target may be used to download coordinates from a host computer. To do this, simply turn off the motor drive outputs with the MF command, then send the string of points in the following format:

1MAxxxx,2MAxxxx,LTm
1MAxxxx,2MAxxxx,LTm+1 .... etc.
**aMA** Move Absolute

This command generates a motion to the absolute position n. The zero, or home position, may be defined by the DH (Define Home) statement. If not defined, it is the position where the controller was when powered on.

n must be in the range of + or - 4,294,976,296 encoder counts. The axis parameter a must be the first character of this command.

Examples: 2MA55000 (rtn) : Axis 2 moves to position 55000
           1MA0 (rtn) : Tells axis 1 to go to Home position

**MC** see EM

**MD** Macro Definition

This command is used to define a new macro command. Any duplication of numbers will simply result in the loss of any previously defined macro using that number.

Examples:

MD4,TT,TP (rtn) : Defines macro #4 as Tell Target, then Tell Position
EM4 (rtn) : Executes macro #4

Since the length of the macro may vary considerably, a new definition of the macro command does not use the same storage space, but occupies the same additional memory space which would be allocated to a macro which had not been previously defined. In other words, repeated use of the same macro number does not save space and is capable of overflowing the assigned memory if overdone. The memory space occupied by the erased macros is not recovered until an RM (Reset Macro) command is issued.

**aMF** Motor OFF

When this command is issued, the motor is no longer held in position control and may be moved by hand. The MF command is used to prevent unwanted movement or to allow for manual positioning of the unit. When manually positioned, however, the motor position is still detected in the MF status and may be reported by the TP command. This means on-line control of manual motor movement is possible.

The opposite command is MN (Motor ON). Use caution when turning the motor back on. The C-808 remembers its position when it received the MF command and it will try to return there unless the target position is redefined.

Examples: 1MF (rtn) : Sets motor 1 OFF
           MF (rtn) : Sets both motors OFF

To set the manually selected position as the new target position, use the DH (Define Home) or the AB (Abort) command before setting the motor back in the MN (Motor ON) state as follows:

2AB,2MN (rtn)
1AB,MN (rtn)

**M** Move To Point Incrementing

This command is used for moving to a previously stored set of coordinates. The first point—to be moved to—is specified by n. Each time this command is executed in a repeat loop, it will advance to the next higher count stored point. Points are
stored with the LP (Learn Point), LI (Learn Incrementing) and LT (Learn Target) commands.

Example: MI7, WS77, WA500, RP5 (rtn)

In this example, the controller will move to points 7, 8, 9, 10, 11 and 12, stopping for 500 ms at each point. Remember that each point stores a position for each axis. If you do not want a given axis to move, you must turn off the motor for that axis before issuing this command.

The MI command may be used for repeating fixed motion patterns at various reference positions. Any number of coordinates may be stored up to the capacity of available memory. Each 2-dimensional coordinate point stored uses 20 bytes of memory. Up to 2560, 2-dimensional points may be stored under normal conditions.

aMN Motor ON

This is the normal system mode, where the C-808 controls the axis position continuously. Each difference in actual and target position causes a movement toward the target.

Example: 2MN (rtn)

Again, use caution when turning the motor back on. The C-808 remembers its position when it received the MF command and it will try to return there unless the target position is redefined.

MPn Move To Point n

Similar to the MI command, but it is used for moving to a single point.

aMRn Move Relative

This command generates a motion of relative distance of n counts in the specified direction from the actual motor position. N may be either a positive or negative number in the range of up to approximately 4 billion (n < 2 exp 32) counts.

The axis parameter a must be specified in the command string.

Examples: 2MR5000 (rtn): Motor 2 moves 5,000 counts

1MR-330 (rtn): Motor 1 moves 330 counts in negative direction

1MR2000, 2MR-1200 (rtn)

PSn Print String

Any previously defined string can be displayed with this command. In addition to those strings defined by the user, it is possible for advanced users to be able to store strings permanently in EPROM. Contact the factory for information on this feature.

RG Reset String Storage

Like macro commands, strings have variable lengths, so the redefinition of a string does not release its storage space. All of the allocated space can be used up by continually redefining the same string. To recover lost string space, it is necessary to use the RG command.

RM Reset Macro
Used only to initialize the memory reserved for macro commands. It clears the macro storage.

Example: RM (rtn) Removes all stored macros

**RPn** Repeat

This command causes the command string to repeat n times. If n is not specified, the commands are repeated 65,535 times. The repeat loop may be interrupted by transferring any character. This character may not be the first character of a new command because it will be discarded.

Example:

1TP,WA500,RP99 (rtn)

Will display the position of axis 1 every 0.5 second for 100 times.

**RS** Reset Point Storage

This command clears the point storage. Unlike the macro command storage, the point storage has a fixed format which means that the points may be defined as often as possible without danger of overflowing the allocated space.

**RT** Reset System

Performs a complete restart of the system, including restoration of all default conditions and DIP switch settings. Both axes are set in the MF condition. It should be used with care if you have made many changes since powering up the controller.

**aSAN** Set Acceleration

Defines the acceleration rate for a given axis in encoder counts/second/second. The value for n may be within the range from 1 to 8,388,608.

**aSDn** Set Deceleration

Defines the deceleration rate for a given axis in encoder counts/second/second. Deceleration is started when the distance to the target is less than the value entered by the SP (Set Point) or CP (Calculate Point) commands. The value n may be within the range from 1 to 8,388,608.

**aSEn** Set Dynamic Gain \((0 < n < 255)\)

This command sets the slope of the proportional relationship between the dynamic position error (during motion) and the motor voltage. If the motor deviates from the position / timeline specified by the velocity setting, the value of the dynamic gain will determine the stiffness of the control loop attempting to zero the error. This loop is updated every 4 ms. If the dynamic gain is set too high or too low the motor may oscillate and vibrate during motion. For encoders with up to 100 lines (400 pulses / rev.) the default value of 32 is OK. For encoders with 500 lines (2,000 pulses / rev.) a value between 6 and 9 is good. The exact number depends on the complete system, friction, inertia etc.

Examples:

SE20 (rtn): Sets both axes to a dynamic gain of 20

2SG7 (rtn): Sets axis 2 to a dynamic gain of 7
aSF  Slave OFF
This is the opposite command to SN (Slave ON). It removes the slave condition from axis [a]. It is not necessary to know or to specify which axis it was slaved to, if any.

aSGn  Set Gain  \(0 < n < 255\)
This command sets the slope of the proportional relationship between the position error and the motor voltage. The higher the gain value is set, the higher is the stiffness of position coupling, so that a small error value causes a proportionally larger motor current driving the motor towards the target.

The default gain value is 32—which is usually stable. The optimum value depends on friction, inertia, motor power, and the resolution of the encoder. It must be determined by the user. Here is some help for the different encoder types offered by PI:

<table>
<thead>
<tr>
<th>Resolution of Encoder</th>
<th>Gain Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000 counts</td>
<td>10 . . . 20</td>
</tr>
<tr>
<td>400 counts</td>
<td>32 . . . 70</td>
</tr>
<tr>
<td>60 counts</td>
<td>50 . . . 100</td>
</tr>
</tbody>
</table>

If the error reported by an axis after completing its motion is excessive, the gain value may be increased in small increments until the error is reduced. If the axis becomes unstable and begins to oscillate, the gain must be reduced until the oscillation stops.

Example:
- SG20 (rtn) : Sets all axes to a gain of 20
- 2SG50 (rtn) : Sets axis 2 to a gain of 50

aSNn  Slave ON
Forces axis a to drive to the same position as axis n. May be used to lock multiple axes together to accomplish a panel control, such as a trackball or mouse. Whenever the position of axis n changes, the target position for axis a is updated to match it.

Example:
- 2SN1 (rtn) : Causes axis 2 to move to the same position as axis 1.

aSON  Set Output to Level n
This command allows the user to manually set the output voltage. It is intended for use with Piezoelectric Translators controlled by the analog signal via an amplifier, like the C-802 series combi controller which uses the C-808 module inside to control PZT Translators. Other devices which require an amplified voltage with 16-bit resolution can be controlled using the 'SO' command (the 'SO' command works identically like the 'SU' command used with the E-255 Control Interface).

When this command is used, the servo loop for axis a is turned off until a reset or some form of move command is generated, such as MR or GH. Some care must be exercised to not use this command with a motor connected, if uncontrolled motion can cause damage to equipment or personnel.

The following diagram shows the relation between the programmed SO (Set Output) value and the actual output voltage:
Output Voltage set by 'SO' Command

The command uses binary two's-complement notation to the digital-to-analog converter. Therefore, a command of 32767 counts will give a maximum positive output, but 32768 will give a maximum negative output. Since it is entirely possible that a user might want to take advantage of this mathematical peculiarity, there is no built-in safeguard against it.

<table>
<thead>
<tr>
<th>Command</th>
<th>Output Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;SO0&quot;</td>
<td>0.0 V</td>
</tr>
<tr>
<td>&quot;SO10000&quot;</td>
<td>+ 3.1 V</td>
</tr>
<tr>
<td>&quot;SO20000&quot;</td>
<td>+ 6.2 V</td>
</tr>
<tr>
<td>&quot;SO32767&quot;</td>
<td>+10.2 V</td>
</tr>
<tr>
<td>&quot;SO32768&quot;</td>
<td>-10.2 V</td>
</tr>
</tbody>
</table>

Example: 2SQ20,MR-999999,WS120,AB,MN,SQ127,DH,TP (rtn)

This compound command causes the system to set the available drive to a value low enough to allow driving into a stop without damage, sets the target value far beyond the total length of the stage and waits for the stop to be encountered. It then turns off the motor, adjusts the target point to be the current position, turns the motor back on, restores full power to be available to the axis, defines the home position to be the position at which it stopped, and signals completion of all this to the operator by reporting the position.

aSPn Set Deceleration Point

Defines the point at which to begin the deceleration phase. It specifies the number of encoder counts from the position being moved to. The value for n may be in the range from 0 to 65,535.

aSQn Set Torque (0 < n < 127)

Set Torque defines the maximum voltage to be applied to the motor while attempting to maintain position or velocity.

Example: 2SQ20,MR-999999,WS120,AB,MN,SQ127,DH,TP (rtn)

Stop Motor
Stop Motor is functionally equivalent to the MF (Motor OFF) command. It turns off the position control loop and the motor may be moved by external forces.

The motion may be resumed by issuing the GO or MN (Motor ON) command.

Example:  
```
ST (rtn)
2ST (rtn)
```

**aSUn Set Velocity**

Causes all subsequent moves to run at the velocity \( n \). The value is given in encoder counts per second. If the torque load changes on the motor, the controller attempts to maintain the velocity by varying the motor current. The value \( n \) may be in the range from 0 to 8,388,608.

Example:
```
1SV24000 (rtn): Sets the velocity of motor 1 to 24,000 counts / second.
```

Related Commands:  
- TY (Tell the programmed velocity)
- TV (Tell actual velocity).

**TAn Tell A/D**

The C-808 controller has four analog input channels which may be used for general measurement applications. Each channel is read by an 8-bit analog-to-digital converter and the digitized value may be reported by the TA command.

The reported number has a value between 0 and 255 and represents the ratio of the analog reading to the reference input. The reference input is +5 Volts.

Examples:
```
Command                Report
TA1 (rtn)              "01A255" CRLF ETX
TA (rtn)               "01A255" CRLF
"02A255" CRLF
"03A255" CRLF
"04A255" CRLF ETX
```

**TCn Tell Channel**

Reports the on/off status on each TTL I/O line. This data is reported separately for each channel (up to 8) but is combined into a single coded value if channel 0 is specified or no channel is specified.

Examples:
```
Command                Report
TC (rtn)               "0255" CRLF
"0255" CRLF ETX
TC2,TC5,TC1 (rtn)      "02C0002" CRLF ETX
"05C0032" CRLF ETX
"01C0000" CRLF ETX
```
aTE  Tell Error
Reports the position error of axis [a] as determined by subtracting the actual position from the target position. This command may be given while the motor is moving. It can be used to determine if the motor is actually moving, if it is moving in the proper direction, and if it is maintaining the position without oscillation.
Example: 1TE,WA150,RP (rtn)

This example will track the calculated positional error of axis #1 during and after the move. Every 150 ms the actual distance from the target is reported. The output may be stopped by entering any character.
Report: “01E0000000000” CRLF ETX

aTF  Tell Following Error
Reports the difference, in encoder counts, between the actual position of the stage and the ideal instantaneous position, based on the programmed values for acceleration, velocity and deceleration.
Report string format “01F0000000000” CRLF ETX

aTG  Tell Gain
Reports the values defined in the SG (Set Gain) command.
Report: “01G0032” CRLF ETX

Tl  Tell Iterations
This command reports the state of the repeat counter. It is useful for determining the number of times a repetitive action has taken place.
Example:
If macro command #4 contains "1MR100,WE32,WA250,RP99", and we transfer XN2,MC4,Tl, then axis 1 will make repetitive moves of 100 steps, with a delay of .25 seconds between steps, until I/O channel 2 becomes false. It will report the state of the counter to the communications channel in use at the time.

TMn  Tell Macros  (0 < n < 128)
Displays all previously stored macro commands. If n = 0 or, if n is not specified, all macros will be displayed (except for macro zero which uses a separate command, TZ). Since macros may be defined in any sequence, the TM command is useful for confirming the existence of, or contents of, macro commands.

aTO  Tell Output
This command reports the present setting of the manually set output voltage for axis a. It is especially useful in situations where an SO command is followed by an unknown number of I/O commands.
Example: XN3,2I01,TO
This command sequence might be used in calibrating an instrument that causes a switch to open (or close) when the correct voltage is applied. It will increase the voltage in steps of one count until the signal connected to I/O channel 3 goes to a logical low level, then it will report the last voltage commanded

aTP  Tell Position
Tell Position reports the absolute position of axis a. TP may be used to monitor motion during both motor on and motor off status.

Command: 2TP,WA100,RP (rtn)

Report: "02P0000005555" CRLF ETX

This command string causes the controller to report the current position every 100 ms.

**aTP**  **Tell Torque**

Tell Torque reports the value programmed with the SQ (Set Torque) command.

Report: "01Q0127"

**aTS**  **Tell Status**

The status command causes the controller to report for each axis a decimal number in the range from 0 to 255. Converted in an 8-bit value, each bit represents a specific status flag of the inquired axis.

Meaning of the Bits: (set==1, not set==0)

| Bit 0 | On Target | Set by Error=0 (exactly!)  
|       |           | Reset by error > 127 or by any move command.
| Bit 1 | Limit Flag | Set if a limit switch is hit.  
|       |           | Reset by next move command.
| Bit 2 | Motor status | Set if motor=OFF  
|       |           | reset if motor=ON,
| Bit 3 | Slew mode | Motor is driven by Pushbutton keyboard  
|       |           | or joystick.
| Bit 4 | Not used |  
| Bit 5 | Slave | Set by slave=on status  
| Bit 6 | Find Edge | Set if find edge mode is active
| Bit 7 | WS | Set if motor stops for more than 128 ms

If you want to detect the end of a move by reading bit 0 of the status byte, be aware that this bit would only be zero if the error is exactly zero. Friction caused by mechanical components may produce a static drag error of a few counts. This may hold this bit at "Off Position" even when the stage has terminated its move. The WE (Wait for Error) command offers a much more useful method for implementing this function.

**aTT**  **Tell Target**

Reports target position. This is the absolute position to which the servo loop will try to drive the motor any time the MN (Motor ON) state is in effect. The target position may be specified directly with the MA (Move Absolute) and several other commands, or indirectly with the MR (Move Relative) command.

If the system is in decimal mode, ten digits will be reported with a leading minus sign (-), if the position is less than the position defined as "home." When the hex mode is in effect, eight digits are reported in two's-complement notation. In both cases, the data is preceded by the axis number and a "T" character and followed by a carriage return and a line feed character.

Examples:

<table>
<thead>
<tr>
<th>Command</th>
<th>Report</th>
</tr>
</thead>
</table>

*Release 4.4  Physik Instrumente (PI)  Page 46*
**TT (rtn)**

"01T00000000000" CRLF
"02T00000000000" CRLF ETX

**aTV**  
**Tell Actual Velocity**
Reports the number of encoder counts received by axis a during the last one second monitoring interval. This command may be used to determine the difference between the actual velocity and the programmed velocity.

Example: 1SV2000,1MR250000,WAS00,1TV (rtn)
Report: "01V0000019998" CRLF ETX

**aTY**  
**Tell Programmed Velocity**
Reports the value programmed with the SV command. The values reported with the TV and TY commands should differ by only a few counts.

Example: 1TY (rtn)
Report: "01Y0000020000" CRLF ETX

**VE**  
**Tell Version**
Reports the revision and release date of the EPROM firmware and the copyright notice.

**WAn**  
**Wait Absolute**
This command inserts a wait period of n milliseconds before going to the next command.

Example: 2MR2000,WAS00,1MR-2000 (rtn)

This command line will move motor 2 2,000 steps, then wait for 3 seconds after the motor starts, then move back 2,000 steps. The wait period of 3 seconds also includes the time the motor moves. If the actual rest time of the motor should be 3 seconds, an additional WS command must be inserted:

2MR2000,3WS77,WAS00,2MR-2000 (rtn)

**aWDn**  
**Wait for Delta Motion n**
This command causes command processing to pause until axis a has moved by n counts relative to the position before the most recent move command. It also generates a pulse on I/O channel 8 (J7-14) when the specified position is reached.

Example: 2MR124345678,MC5 (MD5,2WE1000,RP)
This sequence will cause axis 2 to move toward a position 12345678 encoder counts more positive than the starting position, and will generate a pulse on I/O channel 8 every time it has moved by a delta value of 1,000 counts

**aWEn**  
**Wait for Error**
The WE command causes command processing to pause until axis a has moved to within the error band specified by the value of n. WE is an improved method of
determining the end of motion for a wider range of situations than the WS and TS commands.

The flag bit indicating "on-target" for the axis requires that the error of the axis from the target be exactly equal to zero, and in many cases, there could be one or more counts of error in a stage when it has completed a move. Under these conditions, the system appears to be "hung-up" to first time users.

The WS (Wait for Stop) command can provide similarly unexpected results whenever the gain has been set too high, or there is vibration from an external source. It measures the time that the stage has had absolutely no motion.

Although both these methods are performing the task that they were designed to perform, there are times when they do not fulfill the purpose of the moment. This is what the WE command was designed for—to allow the user to specify the magnitude of error he considers proper for signalling the end of the move.

Example: 1MR12345,WE10,CN6,WA100,CF6,RP999

This sequence will move axis 1 in steps of 12345 counts and will generate a pulse of 100 milliseconds on I/O channel 6 at the end of each stop.

WFn  Wait for Channel OFF
Wait until channel n is at a low level and then continue with next operation.
Example: WF6 (rtn)

WNn  Wait for Channel ON
Wait until channel n is at a low level and then continue with next operation.
Example: WN6 (rtn)

aWPn  Wait for Position n
WP causes command processing to pause until position n has been reached by axis a before proceeding to the next command in line.

When the position is reached, a pulse is generated on I/O channel 8 (J7-14). Please note that this command specifies an absolute position. If the axis never passes this point, the command will wait forever, or until reset.

For relative motion, see the WD command.
Example: GH,WE32,2MA12345,WP12000,1MR54321

This will start axis 2 moving toward position 12,345. When it reaches position 12,000, axis 1 will begin its motion. I/O channel 8 can be used to signal an external process at the same time.

Care must be taken to not use I/O channel 8 for any other purpose when the WP or WD commands are used. Also, note that the position is updated 1,000 times per second, so this command can be no more exact than the number of encoder counts moved in a millisecond. (a velocity of 100,000 counts/second gives an uncertainty of 100 counts.)

aWSn  Wait for Stop
Wait until axis a has stopped moving for n milliseconds before continuing to the next command. The stop condition is true only if no encoder count is detected within the specified n milliseconds.

This command can be used if one motor must perform two motions and must not start the second move before the first move is terminated.
Example: MR5000,1MR-5000 (rtn)
In the example, this command would cause no motor movement, because the target position is increased by 5,000 and decreased by the same amount nearly at the same time. This would take place far quicker than the axis could begin moving.

To perform the desired motion, use the following command:

```
1MR5000,1WS50,1MR-5000
```

The WS command causes the C-808 to wait until motor 1 has stopped for at least 50 ms before continuing on to the command to move it back.

Since this command monitors the motion of the axis, remember to allow enough time for the axis to start moving. This may be done by increasing the value of n or by inserting a WA command before the WS command. Also, remember that very low programmed velocities will not generate an encoder count every millisecond. In practice WS50 . . . WS100 values are recommended.
### Connectors on C-808.00 and C-808.50 Controllers

#### Motor Axis #1

**Connector J1 (10-Pin Flat Ribbon Connector)**

<table>
<thead>
<tr>
<th>PIN</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MOTOR OUT</td>
</tr>
<tr>
<td>2</td>
<td>+ 5 VOLTS</td>
</tr>
<tr>
<td>3</td>
<td>ENCODER CHANNEL A</td>
</tr>
<tr>
<td>4</td>
<td>ENCODER CHANNEL B</td>
</tr>
<tr>
<td>5</td>
<td>ENCODER GROUND (GND)</td>
</tr>
<tr>
<td>6</td>
<td>MOTOR GND</td>
</tr>
<tr>
<td>7</td>
<td>LIMIT SWITCH RIGHT</td>
</tr>
<tr>
<td>8</td>
<td>LIMIT SWITCH LEFT</td>
</tr>
<tr>
<td>9</td>
<td>REFERENCE SIGNAL</td>
</tr>
<tr>
<td>10</td>
<td>LOGIC GND</td>
</tr>
</tbody>
</table>

#### Motor Axis #2

**Connector J2 (10-Pin Flat Ribbon Connector)**

<table>
<thead>
<tr>
<th>PIN</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MOTOR OUT</td>
</tr>
<tr>
<td>2</td>
<td>+ 5 VOLTS</td>
</tr>
<tr>
<td>3</td>
<td>ENCODER CHANNEL A</td>
</tr>
<tr>
<td>4</td>
<td>ENCODER CHANNEL B</td>
</tr>
<tr>
<td>5</td>
<td>ENCODER GND</td>
</tr>
<tr>
<td>6</td>
<td>MOTOR GND</td>
</tr>
<tr>
<td>7</td>
<td>LIMIT SWITCH RIGHT</td>
</tr>
<tr>
<td>8</td>
<td>LIMIT SWITCH LEFT</td>
</tr>
<tr>
<td>9</td>
<td>REFERENCE SIGNAL</td>
</tr>
<tr>
<td>10</td>
<td>LOGIC GND</td>
</tr>
</tbody>
</table>
RS-232 (used with C-808.00)

Connector J3 (10-Pin Flat Ribbon Connector)

<table>
<thead>
<tr>
<th>PIN</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DATA IN (RxD)</td>
</tr>
<tr>
<td>3</td>
<td>DATA OUT (TxD)</td>
</tr>
<tr>
<td>5</td>
<td>Handshake in (see page A16 f)</td>
</tr>
<tr>
<td>7</td>
<td>Handshake out (see page A16 f)</td>
</tr>
<tr>
<td>9</td>
<td>GND</td>
</tr>
<tr>
<td>10</td>
<td>Reset</td>
</tr>
</tbody>
</table>

RS-232 (used with C-808.50)

Connector J5: (96-pin EURO-Connector)

<table>
<thead>
<tr>
<th>PIN</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31c</td>
<td>Data IN (RxD)</td>
</tr>
<tr>
<td>29b</td>
<td>Data OUT (TxD)</td>
</tr>
<tr>
<td>30c</td>
<td>Handshake IN</td>
</tr>
<tr>
<td>28c</td>
<td>Handshake OUT</td>
</tr>
<tr>
<td>5a</td>
<td>5 VDC</td>
</tr>
<tr>
<td>6abc</td>
<td>GND</td>
</tr>
</tbody>
</table>

Joystick (used with C-808.00)

Connector J-6 (16-pin Flat Ribbon Connector)

<table>
<thead>
<tr>
<th>PIN</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+5 VDC</td>
</tr>
<tr>
<td>2</td>
<td>N.C.</td>
</tr>
<tr>
<td>3</td>
<td>JOYSTICK SWITCH #1</td>
</tr>
<tr>
<td>4</td>
<td>N.C.</td>
</tr>
<tr>
<td>5</td>
<td>ANALOG INPUT #1</td>
</tr>
<tr>
<td>6</td>
<td>N.C.</td>
</tr>
<tr>
<td>7</td>
<td>GND</td>
</tr>
<tr>
<td>8</td>
<td>N.C.</td>
</tr>
<tr>
<td>9</td>
<td>N.C.</td>
</tr>
<tr>
<td>10</td>
<td>N.C.</td>
</tr>
<tr>
<td>11</td>
<td>ANALOG INPUT #2</td>
</tr>
<tr>
<td>12</td>
<td>N.C.</td>
</tr>
<tr>
<td>13</td>
<td>JOYSTICK SWITCH #2</td>
</tr>
<tr>
<td>14</td>
<td>N.C.</td>
</tr>
<tr>
<td>15</td>
<td>N.C.</td>
</tr>
<tr>
<td>16</td>
<td>N.C.</td>
</tr>
</tbody>
</table>

Joystick (used with C-808.50)

<table>
<thead>
<tr>
<th>96-pin Connector</th>
<th>Joystick</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN #</td>
<td>PIN #</td>
</tr>
<tr>
<td>5 abc</td>
<td>+5 VDC OUT</td>
</tr>
<tr>
<td>6 abc</td>
<td>GND</td>
</tr>
<tr>
<td>15 b</td>
<td>Analog input #2</td>
</tr>
<tr>
<td>15 c</td>
<td>Analog input #1</td>
</tr>
<tr>
<td>22 a</td>
<td>Joystick switch #2</td>
</tr>
<tr>
<td>23 c</td>
<td>Joystick switch #1</td>
</tr>
</tbody>
</table>

Joystick switches (pushbuttons) have to be pulled up to +5 Volt via 10 kOhm resistors.

DIP Switch connection using C-808.50

If you want to connect an external DIP switch to the 96 pin connector, an additional 74ALS746N has to be used to latch the settings.

More information on request.
### I/O Connector (used with C-808.00)

**Connector J7 (16-Pin flat ribbon connector)**

<table>
<thead>
<tr>
<th>PIN</th>
<th>Description</th>
<th>DSUB15-pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LOGIC GND</td>
<td>(1)</td>
</tr>
<tr>
<td>2</td>
<td>-5 VOLTS</td>
<td>(2)</td>
</tr>
<tr>
<td>3</td>
<td>RESERVED (DO NOT CONNECT)</td>
<td>(3)</td>
</tr>
<tr>
<td>4</td>
<td>RESERVED (DO NOT CONNECT)</td>
<td>(4)</td>
</tr>
<tr>
<td>5</td>
<td>RESERVED (DO NOT CONNECT)</td>
<td>(4)</td>
</tr>
<tr>
<td>6</td>
<td>RESERVED (DO NOT CONNECT)</td>
<td>(5)</td>
</tr>
<tr>
<td>7</td>
<td>I/O CHANNEL 1</td>
<td>(6)</td>
</tr>
<tr>
<td>8</td>
<td>I/O CHANNEL 2</td>
<td>(7)</td>
</tr>
<tr>
<td>9</td>
<td>I/O CHANNEL 3</td>
<td>(8)</td>
</tr>
<tr>
<td>10</td>
<td>I/O CHANNEL 4</td>
<td>(9)</td>
</tr>
<tr>
<td>11</td>
<td>I/O CHANNEL 5</td>
<td>(10)</td>
</tr>
<tr>
<td>12</td>
<td>I/O CHANNEL 6</td>
<td>(11)</td>
</tr>
<tr>
<td>13</td>
<td>I/O CHANNEL 7</td>
<td>(12)</td>
</tr>
<tr>
<td>14</td>
<td>I/O CHANNEL 8</td>
<td>(13)</td>
</tr>
<tr>
<td>15</td>
<td>+5 VOLTS</td>
<td>(14)</td>
</tr>
<tr>
<td>16</td>
<td>N.C.</td>
<td>(15)</td>
</tr>
</tbody>
</table>

### I/O Connector (used with C-808.50) (96-pin Connector)

**PIN #**

<table>
<thead>
<tr>
<th>PIN #</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1aabc</td>
<td>+12 Volts</td>
</tr>
<tr>
<td>2aabc</td>
<td>+12 Volts</td>
</tr>
<tr>
<td>5abc</td>
<td>+5 Volts out</td>
</tr>
<tr>
<td>6abc</td>
<td>Logic GND</td>
</tr>
<tr>
<td>25a</td>
<td>I/O Channel #7</td>
</tr>
<tr>
<td>25b</td>
<td>I/O Channel #8</td>
</tr>
<tr>
<td>26a</td>
<td>I/O Channel #4</td>
</tr>
<tr>
<td>26b</td>
<td>I/O Channel #5</td>
</tr>
<tr>
<td>26c</td>
<td>I/O Channel #6</td>
</tr>
<tr>
<td>27a</td>
<td>I/O Channel #1</td>
</tr>
<tr>
<td>27b</td>
<td>I/O Channel #2</td>
</tr>
<tr>
<td>27c</td>
<td>I/O Channel #3</td>
</tr>
</tbody>
</table>

### Main Connector on C-808.50

**96-Pin CONNECTOR**

<table>
<thead>
<tr>
<th>Pin #</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>-MOTOR SUPPLY</td>
</tr>
<tr>
<td>A2</td>
<td>+MOTOR SUPPLY</td>
</tr>
<tr>
<td>A3</td>
<td>LCD LNA</td>
</tr>
<tr>
<td>A4</td>
<td>nc</td>
</tr>
<tr>
<td>A5</td>
<td>+5VDC</td>
</tr>
<tr>
<td>A6</td>
<td>GND</td>
</tr>
<tr>
<td>A7</td>
<td>PAL-15</td>
</tr>
<tr>
<td>A8</td>
<td>nc</td>
</tr>
<tr>
<td>A9</td>
<td>nc</td>
</tr>
<tr>
<td>A10</td>
<td>nc</td>
</tr>
<tr>
<td>A11</td>
<td>nc</td>
</tr>
<tr>
<td>A12</td>
<td>A2</td>
</tr>
<tr>
<td>A13</td>
<td>A5</td>
</tr>
<tr>
<td>A14</td>
<td>PHO</td>
</tr>
<tr>
<td>A15</td>
<td>RD</td>
</tr>
<tr>
<td>A16</td>
<td>A15</td>
</tr>
<tr>
<td>A17</td>
<td>A13</td>
</tr>
<tr>
<td>A18</td>
<td>A9</td>
</tr>
<tr>
<td>A19</td>
<td>AD7 (SW1-8)</td>
</tr>
<tr>
<td>A20</td>
<td>AD4 (SW1-5)</td>
</tr>
<tr>
<td>A21</td>
<td>AD1 (SW1-2)</td>
</tr>
<tr>
<td>A22</td>
<td>P3-7</td>
</tr>
<tr>
<td>A23</td>
<td>P3-4</td>
</tr>
<tr>
<td>A24</td>
<td>P3-1</td>
</tr>
<tr>
<td>A25</td>
<td>I/O #7</td>
</tr>
</tbody>
</table>

Release 4.4 Physik Instrumente (PI) Page 52
C-808 DC Motor Controller Operating Manual MS 33E

<table>
<thead>
<tr>
<th>A26</th>
<th>I/O #4</th>
<th>B26</th>
<th>I/O #5</th>
<th>C26</th>
<th>I/O #6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A27</td>
<td>I/O #1</td>
<td>B27</td>
<td>I/O #2</td>
<td>C27</td>
<td>I/O #3</td>
</tr>
<tr>
<td>A28</td>
<td>U6-11</td>
<td>B28</td>
<td>U6-5</td>
<td>C28</td>
<td>U8-3</td>
</tr>
<tr>
<td>A29</td>
<td>U6-13</td>
<td>B29</td>
<td>U8-2</td>
<td>C29</td>
<td>U8-10</td>
</tr>
<tr>
<td>A30</td>
<td>U7-14</td>
<td>B30</td>
<td>nc</td>
<td>C30</td>
<td>U7-9</td>
</tr>
<tr>
<td>A31</td>
<td>U6-3</td>
<td>B31</td>
<td>U7-6</td>
<td>C31</td>
<td>U7-15</td>
</tr>
<tr>
<td>A32</td>
<td>U7-1</td>
<td>B32</td>
<td>U7-2</td>
<td>C32</td>
<td>U7-7</td>
</tr>
</tbody>
</table>

**Power Connector (used on C-808.00)**
Connector JB (4-Position Power Connector)

| PIN 1: | +12 VDC to +40 VDC (MOTOR SUPPLY) |
| PIN 2: | -12 VDC to -40 VDC (MOTOR SUPPLY)  |
| PIN 3: | GND                                |
| PIN 4: | 5 VDC                              |

**Power Connection (used with C-808.50)**
96-pin Connector

| PIN 1abc: | -12 VDC to -40 VDC (MOTOR SUPPLY), Standard: -15 V |
| PIN 2abc: | +12 VDC to +40 VDC (MOTOR SUPPLY), Standard: +15 V |
| PIN 5abc: | 5 VDC                                         |
| PIN 6abc: | GND                                          |
**Physik Instrumente (PI) GmbH & Co.**

Polytec-Platz 1–7, D-76337 Waldbronn
Germany
Tel. (0 72 43) 604 100
Fax (0 72 43) 604 145
Web page: www.physikinstrumente.com
Email: info@physikinstrumente.com

**Sales Office Berlin:**
Polytec PI
Rudower Chaussee 5
D-12489 Berlin
Tel. (0 30) 63 92 51 40
Fax (0 30) 63 92 51 41

**Germany:**
Pl Ceramic GmbH
Lindenstrasse D-07389 Lederhose
Tel. (0 36 66) 04 8 82- 0
Fax (0 36 66) 04 8 82-25

**PI International**

**USA:**
Polytec PI, Inc.
1342 Bell Ave., Suite 3 A
Tustin, CA 92780
Tel. (714) 850 1835
Fax (714) 850 1831

**Japan:**
Polytec KK
Akabono-cho 2-38-5
Tachikawa-shi
J-Tokyo 190
Tel. (0 42) 26 73 00
Fax (0 42) 26 73 01

**UK:**
Lambda Photometrics, Ltd.
Lambda House, Batford Mill
GB-Harpenden, Herts AL5 5BZ
Tel. (0 15 82) 76 43 34
Fax (0 15 82) 71 20 84

**Italy:**
Physik Instrumente (PI) S.r.l.
Via E. De Amicis, 2
I-20091 Brusso (MI)
Tel. (02) 665 011 01
Fax (02) 665 014 56

**Product Line**

**Fiber Aligners and Positioners**
- Automatic search for "hot spot".
- 6x6x6 mm travel range with 0.01 μm resolution.
- 1D, 2D and 3D scan functions for device characterization.

**Piezo Flexure Stages**
- Displacement range up to 400 μm.
- Better than 1 nanometer resolution.
- Wire EDM cut flexures for zero friction.
- Single and multi-axis systems.
- OEM and custom designs.

**Piezo Tilting Platforms**
- Up to three axes, sub-arc-second resolution.
- Tilt range up to 5 milliradians.
- Sub-millisecond response.

**Piezo Actuators**
- Widest range of PZT actuators world-wide.
- Stack actuators: 5 to 200 μm displacement.
- Multilayer technology.
- Sub-nm resolution, sub-millisecond response.
- Integrated position sensors.
- Bimorphs, benders, tubes, lever amplified actuators.
- PZT ceramics and dielectric ceramics.
- Amplifiers, servo controllers, computer interfaces ...
- OEM and custom designs.

**Capacitive Displacement Sensors**
- Better than 0.1 nanometer resolution.
- Measuring ranges: 15 to 300 μm.
- Better than 0.05% linearity.
- Up to 10 kHz bandwidth.

**Linear/Rotary Micropositioners**
- 6 to 300 mm travel, resolution better than 0.1 μm.
- DC servo motor- and stepper motor drives.
- Rotary stages: resolution better than 0.001 degrees.
- Optional high resolution PZT drives.
- Fast voice-coil driven stages (250 mm/s).
- Hexapod: Six-degrees-of-freedom micropositioner.

**Motor Controllers**
- DC servo- and 5 phase stepper motor controllers.
- Bench-top and PC plug-in-board versions.

**Optical Tables/Vibration Isolation**
- Steel-honeycomb and stone tables.
- Pneumatic isolators with automatic height adjustment.
- Work stations with arm rests and shelves.

**Opto-Mechanical Components**
- X-profile benches.
- Precision holders for mirrors, lenses, prisms, beam splitters, etc.
- Gimbal mounts with sub-arcsec resolution PZT drives.