



**User Manual      MS 67E**

***Mercury***  
C-860.10

**Single-Axis DC-Motor Controller**

V 4.00

---

---

***Technical Documentation  
&  
User Instructions***

---

---

**This Document is valid for the following Product:**

**C-860.10    *Mercury* DC-Motor Controller**  
(using firmware version 6.10)

---

---

*Release:*            4.00  
*Release Date:*    2001-10-08

---

---

---

© **PI** (Physik Instrumente) GmbH & Co.  
76228 Karlsruhe, Germany    FAX: (+49)721-4846-100

---

E-Mail: [info@pi.ws](mailto:info@pi.ws)

[www.pi.ws](http://www.pi.ws)

---

## Table of Contents :

|           |   |           |
|-----------|---|-----------|
| <b>0.</b> | <b>MANUFACTURER DECLARATIONS .....</b>        | <b>3</b>  |
| 0.1.      | Declaration of Conformity.....                | 3         |
| 0.2.      | Warnings and Safety Instructions .....        | 3         |
| 0.3.      | Quality and Warranty Clauses.....             | 4         |
| <b>1.</b> | <b>INTRODUCTION .....</b>                     | <b>5</b>  |
| 1.1.      | Product Survey and Contents of Delivery ..... | 6         |
| 1.2.      | Front and Back Panel Elements .....           | 7         |
| <b>2.</b> | <b>SYSTEM DESCRIPTION.....</b>                | <b>8</b>  |
| 2.1.      | Firmware Version Notes .....                  | 8         |
| 2.2.      | Multi-Axis Control.....                       | 8         |
| 2.3.      | LED Status Indicators.....                    | 9         |
| 2.4.      | Default Values.....                           | 9         |
| 2.5.      | Cable Connections.....                        | 10        |
| 2.6.      | Limit Sensors .....                           | 10        |
| 2.7.      | Position Reference Sensors .....              | 11        |
| 2.8.      | Motor Connector Pin Assignment .....          | 11        |
| <b>3.</b> | <b>BEGINNING OPERATION.....</b>               | <b>12</b> |
| 3.1.      | Jumper Settings .....                         | 12        |
| 3.1.1.    | Open Mercury for jumper setting .....         | 13        |
| 3.2.      | RS-232 Communication .....                    | 14        |
| 3.3.      | Networking .....                              | 14        |
| 3.3.1.    | Setting the Network Address .....             | 14        |
| 3.3.2.    | Address Code.....                             | 15        |
| 3.3.3.    | Enabling Communication Upon Power-up .....    | 15        |
| 3.4.      | Communication .....                           | 16        |
| 3.4.1.    | Test Communication .....                      | 16        |
| 3.4.2.    | Setting Motion Control Parameters.....        | 16        |
| 3.4.3.    | Operating Motors and Stages.....              | 17        |
| <b>4.</b> | <b>COMMAND TYPES .....</b>                    | <b>18</b> |
| 4.1.      | Single Commands.....                          | 18        |
| 4.2.      | Compound Commands.....                        | 18        |
| 4.3.      | Macro Commands .....                          | 19        |
| 4.4.      | Macro Zero.....                               | 20        |
| 4.5.      | Reporting Commands .....                      | 20        |
| <b>5.</b> | <b>SOFTWARE .....</b>                         | <b>21</b> |
| 5.1.      | Host Software.....                            | 21        |
| 5.2.      | Windows Library (DLL) .....                   | 25        |
| 5.3.      | LabView Driver Library .....                  | 25        |
| <b>6.</b> | <b>TROUBLE SHOOTING.....</b>                  | <b>25</b> |
| <b>7.</b> | <b>COMMAND REFERENCE .....</b>                | <b>26</b> |
| 7.1.      | MOTION and SEQUENCING COMMANDS .....          | 28        |
| 7.2.      | PARAMETER SETUP COMMANDS .....                | 31        |
| 7.3.      | REPORTING COMMANDS.....                       | 33        |
| 7.4.      | MACRO COMMANDS.....                           | 38        |
| 7.5.      | UTILITY COMMANDS.....                         | 38        |

## 0. Manufacturer Declarations

### 0.1. Declaration of Conformity

The manufacturer,

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



declares, that the **Mercury** Controller, as described in this operating manual, conforms with European standards as follows:

EMC: EN55022 (1991), Group 1, Class B

EN50082-1 (1992) / IEC 801-4: 1988 (1 kV power lines, 0.5 kV Signal lines)

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

### 0.2. Warnings and Safety Instructions



The **Mercury** controller outputs PWM control signals capable of driving high-power external amplifiers. Be aware that mechanical stages may cause damage to persons or property if the controller is not kept under proper software and hardware control.

Take special care when connecting products from other manufacturers. Always follow the General Accident Prevention Rules!

© Copyright 2001 by Physik Instrumente GmbH

Release: 4.00

File:MS67E400.doc, 566784 Bytes

PDate: 08.10.01 11:47

### 0.3. Quality and Warranty Clauses

#### Certification

**P I** (Physik Instrumente) certifies that this product met its published specifications at the time of shipment. The device was calibrated and tested in the same configuration as shipped.

#### Warranty

This **P I** product is warranted against defects in materials and workmanship showing up within a period of one year from date of shipment. Duration and conditions of warranty for this product may be superseded when the product is integrated into (becomes a part of) another Physik Instrumente product. During the warranty period, Physik Instrumente will, at its option, either repair **or** replace products with defects covered by warranty.

#### Limitation of Warranty

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

The design and implementation of any circuitry containing this product is the sole responsibility of the buyer. **PI's** warranty does not cover the buyer's circuitry or any **PI** product malfunction that may result from said circuitry. In addition, **PI** cannot be responsible for any damage that may occur as a result of the buyer's circuitry or for any defects or damage that may result from the use of buyer-supplied products.

**No other warranty is expressed or implied. Physik Instrumente explicitly denies any warranty of merchantability or fitness for any particular purpose.**

#### Disclaimer of Responsibility

**P I** does not assume any responsibility for use of any circuitry or software described in this manual, nor does it make any guarantee as to the accuracy of this manual. **P I** reserves the right to change the product specifications and the functionality of software products and software tools, or the manual itself, at any time.

## 1. Introduction

This manual is provided as an aid in operating the **Mercury** DC-Motor Controller.

The **Mercury** is a miniaturized servo-controller intended for motion control in research and industrial applications. It provides, in a single package, a complete stand-alone control system for the smaller motors typically used in high-precision positioning systems.

The **Mercury** utilizes quadrature encoder signals for position feedback. Depending on the resolution of the encoder scale, incremental resolutions of 0.1 micrometer can be achieved.

The **Mercury** provides PID servo-control of position, velocity, and acceleration. Although each parameter is programmable, each is also set to a programmable default value upon power-up.

Using the built-in commands and features, the user will very quickly be able to command the motor to move to any desired position. During the move, the velocity and acceleration will be controlled in accordance with the most recent settings of the corresponding parameters—it is not necessary to set them up anew for each move.

**Mercury** is designed to operate motorized PI stages. The starter package comes with all connecting cables, power supply and software necessary for immediate operation.



C-860.10 *Mercury* Controller

## 1.1. Product Survey and Contents of Delivery

| Product         | Contents of Delivery  |
|-----------------|---|
| <b>C-860.00</b> | <i>Mercury Controller Set</i> , (address set to 0), including<br>Mercury Controller (order# C-860.10)<br>Power Supply (order# C-890.P12)<br>RS-232 Cable (order# C-815.34)<br>Software (order# C-860.TO)  |
| <b>C-860.10</b> | <i>Mercury Controller</i> (address as specified in order), including<br>RS-232 Cable (order# C-815.34)<br>Software (order# C-860.TO)  |
| <b>C-860.S1</b> | <i>Mercury Controller Set</i> , (address set to 1), including<br>Mercury Controller (order# C-860.10)<br>Power Supply (order# C-890.P12)<br>Network Cable (order# C-860.CN)<br>Software (order# C-860.TO) |
| <b>C-860.S2</b> | same as C-860.S1, but address set to 2  |
| <b>C-860.S3</b> | same as C-860.S1, but address set to 3  |

### Accessories

|                  |   |
|------------------|---|
| <b>C-860.CN</b>  | Network Cable for Mercury to Mercury Daisy-Chain Connection |
| <b>C-860.TO</b>  | Mercury Software Tools                                      |
| <b>C-890.PS</b>  | Mercury Power Supply, 15 VDC                                |
| <b>C-890.P12</b> | Mercury Power Supply, 12 VDC                                |
| <b>C-815.34</b>  | RS-232 Null Modem Cable                                     |

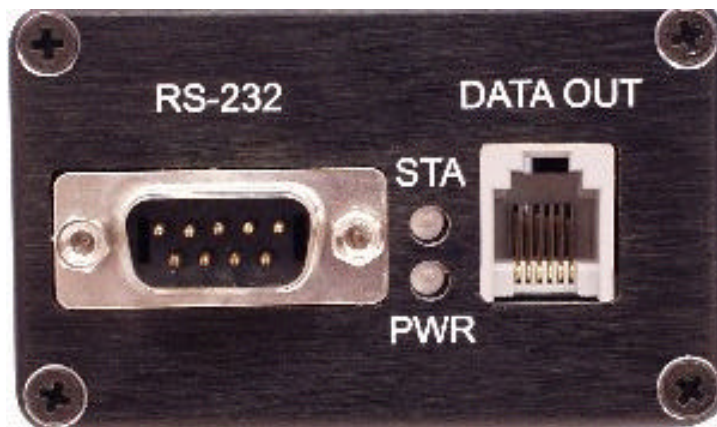
All Mercury Controllers are shipped with Operating Manuals and a Windows Software package. A 12V power supply (C-890.P12) or a 15V power supply (C-890.PS) is included.



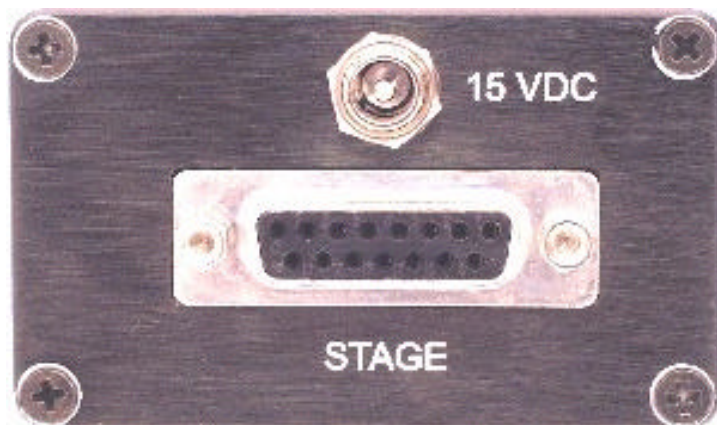
Dwg: MercMike.jpg

*One of many typical applications:  
Mercury Controller with an M-235.5DG Linear Actuator.*

## 1.2. Front and Back Panel Elements



Dwg.: Mercury\_front.jpg



Dwg.: Mercury\_back.jpg

### Connectors :

|                 |              |   |
|-----------------|--------------|---|
| <b>RS-232</b>   | Sub-D 9(m)   | Serial network input                              |
| <b>DATA OUT</b> | RJ11,6c      | Serial network output for next controller, if any |
| <b>STAGE</b>    | DSUB15(f)    | Motor connector                                   |
| <b>15VDC</b>    | Cinch 2.5 mm | Power supply, range +12 to +15VDC                 |

### Indicators:

|            |                         |
|------------|-------------------------|
| <b>STA</b> | LED status, Indicator 1 |
| <b>PWR</b> | LED status Indicator 2  |

## 2. System Description

### 2.1. Firmware Version Notes

#### Version 6.10

The current firmware version is 6.10 (checksum: 50EE F516).

*Changes from version 6.00 to 6.10 are:*

1. Improved WS function
2. After a controller reset communication with the controller is disabled until the controller's valid address code is received. This behavior makes it possible to reset a Mercury working in a network without causing communication errors. This firmware version requires Mercury software version 2.0 or later.?? See the "Host Software" section for more information.
3. Compatibility Note: If using Mercury Windows software version 1.31 with this firmware version, set the start option to "network", otherwise no communication can be established. See the "Host Software" section for more information.

#### Version 6.00 (obsolete)

Checksum version 6.00 is 50EE FB2F

*Any firmware version over 5.xx can be updated to 6.10 by data download.*

### 2.2. Multi-Axis Control

Mercury controllers can be networked.

Up to 16 Mercury Controllers can be connected to one RS-232 port in a daisy chain (option for networking up to 16 controllers on request). The networking feature permits addressing each controller individually. During communication with a controller, all other controllers in the network will stop communication. When desired, communication with the selected controller can be concluded and communication with a different controller initiated.

Switching between controllers requires sending a particular extended ASCII character (0x01) followed by the address number character. This sequence will be referred to as an "address selection command." Each controller compares the address number sent with its own address. If there is a match, the controller enables command interpretation and response so as to respond to subsequent commands. If not, it disables all responses and ignores any subsequent transmissions that are not address selection commands.

The controllers can continue internal macro operation even when no longer addressed. This allows all controllers connected to execute their individual macro commands at the same time. There is no (logical) communication from controller to controller, only from controller to the PC. The host program handles address selection and motion sequencing??. This communication model defines the limits for path interpolation and multi-axis motion control as well as for conditional motion execution.

Any motion sequence or operation begun prior to receiving a disabling address selection command will continue to be executed, except for those that report



data[EC01]. In this manner, each **Mercury** on the bus can be addressed, programmed to execute any desired operation or sequence of operations, then de-addressed. The same or a different command sequence can then be sent to another controller. See the "Networking" section for programming examples.

### 2.3. LED Status Indicators

When power is first applied to the controller, the "STA" LED (upper LED) will glow red. This indicates that the motor servo-loop is disabled, which is the normal state at start-up. The "PWR" LED (lower LED) should be green. This indicates that no command processing error has occurred.

When an MN command is issued, the STA-LED should turn green and remain so until the motor servo-loop is disabled, either by command (such as MF,AB or RT) or until a motion error automatically disables the loop. In any case, an MN command will re-enable the loop and return the LED to the green status unless a problem continues to exist.

If an error is made in transmitting commands to **Mercury**, either manually?? or from a host computer, the PWR-LED will become red until the next valid character is received. If a command is entered without achieving the expected results, check the PWR-LED to see if it is red or green.

There are some conditions where both status LEDs may glow more brightly or may be dark. This indicates an internal busy mode in which communication is suspended.

The following commands cause internal busy times:

UD (busy time about 0.5 seconds),  
RM (busy time about 2 seconds)

### 2.4. Default Values

**Mercury** has been designed to be as easy to use as possible. All motion control parameters, including velocity and acceleration can be permanently stored to be available after the next power-up.

When shipped, Mercury comes with the following (factory) default settings:

|               |          |
|---------------|----------|
| velocity:     | SV6000   |
| acceleration: | SA150000 |
| p-term:       | DP35     |
| i-term:       | DI0      |
| d-term:       | DD0      |
| i-Limit:      | DL2000   |
| echo OFF:     | EF       |
| Limit high:   | LH       |
| Limit on:     | LN       |
| Brake OFF     | BF       |

Note 1: These values can be modified and be stored as power-on defaults at any time.

Note 2: In most cases, the factory default values will not be appropriate for the specific motor-driven mechanics used in your application. It is

recommended that you reset the values to those suggested in the user manual of the mechanics.

Proper motion control parameter settings depend on the individual stage and drive connected. Once the right parameters are found, they can be stored permanently using the UD (update) command.

The UD command takes about 0.5 s to execute. During that time both status LEDs may glow more brightly or may go off. During that time, no communication with the controller will occur.

## 2.5. Cable Connections

There are four connectors on the **Mercury** to provide access to its functions. The motor connections are included in the D-sub 15-pin connector with the encoder and limit switch signals.

The 9-pin sub-D connector is wired straight to the RJ-11 connector. This is done to allow easy "daisy-chaining" from unit to unit. The connections between individual **Mercurys** can be made using the C-860.CN network cables. Note that individual controller addresses must be set up for network operation.

The 2.1 mm jack is the power connection. The motor power (typically +15 V) must be connected to the center pin.

The DSub-15 connector is the motor or stage connection compatible with all PI stages and motor drives. Use the C-815.38 *Connecting Cable* for stage connection.



C-860.CN : Mercury network cable

## 2.6. Limit Sensors

During operation, the limit sensors (or switches) are used to stop the motion at the end of the allowable travel range. Only one of the two switches will interrupt motion in a given direction. If positioning equipment from PI is used, the limit switches or sensors are wired for operation with Mercury. When connecting other motor drives or mechanics, the correct limit switch wiring must be determined. To do this, set the system into a safe position, set a low velocity with the SV command, begin motion with an MR, MA or FE command and enable limit switch operation with the LN command. While the motor is moving, actuate the limit switch toward which it is moving. If the motor stops, that is the correct switch to connect at the end of desired motion for that direction. If the motor does not stop,

actuate the other limit switch. If the motor still does not stop, remove power and read further.

The Mercury can be configured to accept either a high stop signal (+5V) or a low stop signal (ground) from the limit switches (both must be the same). The default is to inhibit movement for the high state. This may be changed with the LL (Limits Low) command.

## 2.7. Position Reference Sensors

Reference signal detection can be used determine precisely the physical origin (zero) position of the stage during system start-up. Since the incremental encoders used for position feedback do not tell you the starting position (all counters are reset during power up), a position reference switch or sensor (origin sensor) in conjunction with a capture mechanism can be used for this purpose.

The FE and FE1 commands are used to start a reference signal search run:

- FE(return)     starts origin search in positive direction
- FE1(return)   starts origin search in negative direction.

The maximum velocity for the reference search is limited to max. 200,000 c/s.

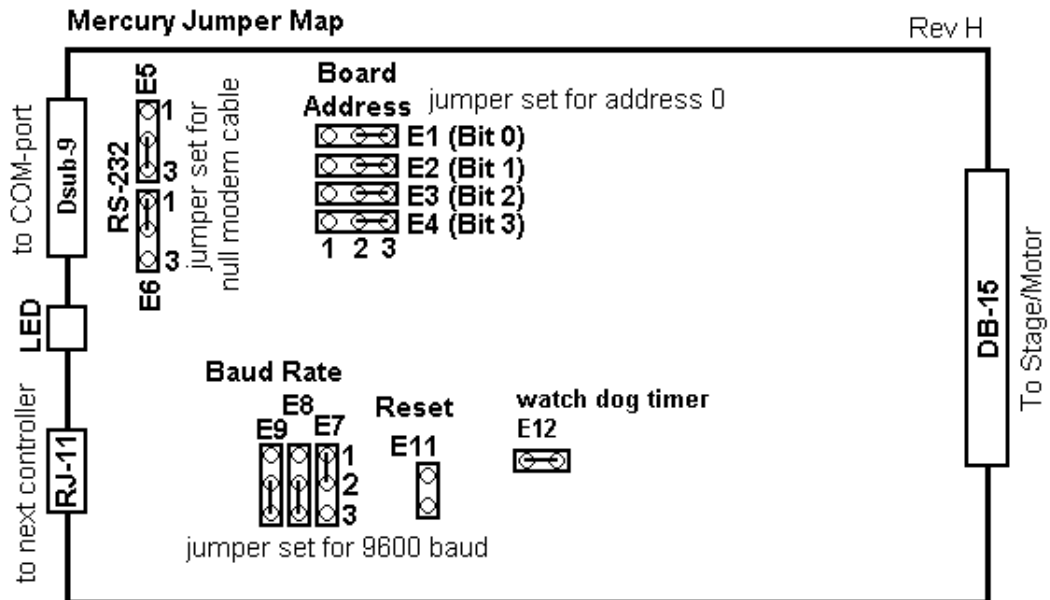
## 2.8. Motor Connector Pin Assignment

Connector type: Sub-D 15(f), "STAGE"

| PIN | Function                      |
|-----|-------------------------------|
| 1   | + 5V (output)                 |
| 9   | Motor ( - ) (output)          |
| 2   | Motor (+) (output)            |
| 10  | Power-GND                     |
| 3   | PWM magnitude (output)        |
| 11  | PWM sign (output)             |
| 4   | +5 V (output)                 |
| 12  | Negative limit (input)        |
| 5   | Positive limit (input)        |
| 13  | Reference signal (input)      |
| 6   | Limit GND                     |
| 14  | Encoder: A(+) / ENCA (input)  |
| 7   | Encoder: A( - ) (input)       |
| 15  | Encoder: B (+) / ENCB (input) |
| 8   | Encoder: B ( - ) (input)      |

### 3. Beginning Operation

#### 3.1. Jumper Settings



Dwg: Merc\_jumperH.tif

#### Controller Address Setting:

(set the jumpers to that side where the shadowed squares are)

#### Address Setting

|    | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|----|---|---|---|---|---|---|---|---|
| E1 |   |   |   |   |   |   |   |   |
| E2 |   |   |   |   |   |   |   |   |
| E3 |   |   |   |   |   |   |   |   |
| E4 |   |   |   |   |   |   |   |   |

| Jumper | Address0 | Address1 | Address2 | Address3 |
|--------|----------|----------|----------|----------|
| E1     | 2-3      | 1-2      | 2-3      | 1-2      |
| E2     | 2-3      | 2-3      | 1-2      | 1-2      |
| E3     | 2-3      | 2-3      | 2-3      | 2-3      |
| E4     | 2-3      | 2-3      | 2-3      | 2-3      |

and so on for addresses up to 15

#### RS-232 Cable Selection :

| Jumper | Null Modem | Straight cable |
|--------|------------|----------------|
| E5     | 2-3        | 1-2            |
| E6     | 1-2        | 2-3            |

*Firmware update mode :*

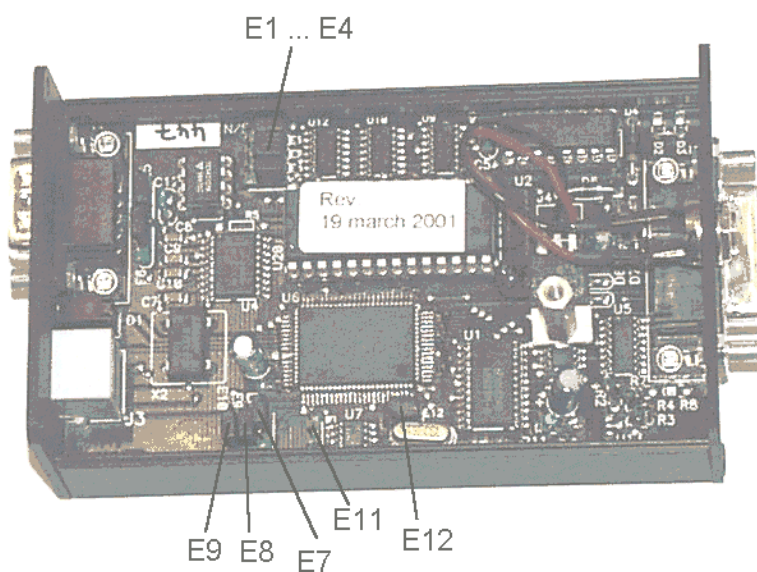
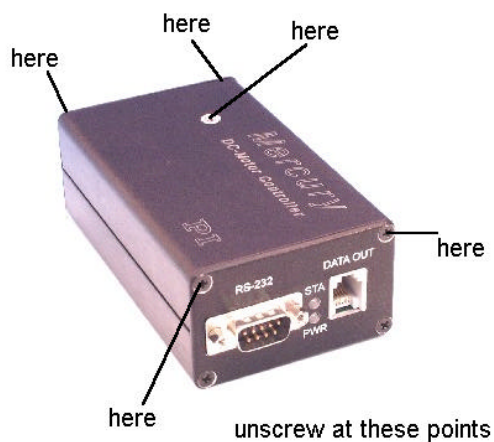
| Jumper | Normal operation | Update Mode |
|--------|------------------|-------------|
| E7     | 1-2              | 2-3         |

*Baud rate Setting :*

| Jumper | 9600 | 19200 |
|--------|------|-------|
| E8     | 2-3  | 1-2   |
| E9     | 2-3  | 2-3   |

*Additional Jumpers :*

|                      |  |
|----------------------|--|
| E11 Reset :          | close (short) pins briefly for hardware reset      |
| E12 Watchdog timer : | installed (shorted) to activate watchdog (default) |

**3.1.1. Open Mercury for jumper**

### 3.2. RS-232 Communication

Serial communication is set by factory to

9600 baud, 8 data, 1 stop, no parity

Internal buffers are used, so there is no handshake required.

Use a null modem cable for connection (order # C-815.34).

### 3.3. Networking

With firmware version 6.10 and later, **Mercury** has to be addressed at least once after power-up or after a reset by command to initiate communication, independent of the address to which the controller is set.

*Note: This behavior is different from that of firmware versions smaller than 6.10, where after a reset the controller was activated and ready to communicate.*

Even if you have only one controller connected, first send an address selection command with the valid address number for the controller in order to initiate communication.

The address for the **Mercury** is set by address jumpers. After sending the address selection code, you can verify the correct activation by sending the TB command. If there is a **Mercury** connected with the same address as the one most recently sent, it will respond to the TB command with the same address. If there is no response, then the address did not match any **Mercury** connected to the RS-232 port.

#### 3.3.1. Setting the Network Address

When ordered, Mercurys are shipped set to these addresses:

Order# C-860.00 : set to address 0

Order# C-860.S1 : set to address 1

Order# C-860.S2 : set to address 2

Order# C-860.S3 : set to address 3

Individual addresses can be set from 0 to 15 by internal jumpers (see the "jumper setting" section).

Each Mercury in the network has to be set to a unique address.

### 3.3.2. Address Code

After power up no controllers with firmware version 6.10 or newer communicate, no matter what the address set. When the first address selection command is sent, the controller with the matching address (if any) will listen for, interpret and respond to subsequent commands as appropriate.

The address selection command consists of 2 characters: 0x01 + AddressCharacter

AddressCharacter = 0x30..0x39 (numerals 0 to 9) , 0x41..0x46 (letters A to F).

*Examples:*

Address selection command for controller with address #0: 0x01+0x30

Address selection command for controller with address #1: 0x01+0x31

Address selection command for controller with address #2: 0x01+0x32

### 3.3.3. Enabling Communication Upon Power-up

As default, Mercury does not communicate until it is addressed.

There are situations where Mercury is to be used with a terminal device which is unable to send address selection command characters. In this case, Mercury can be programmed for immediate communication after power-up or reset.

To do so, use a terminal which *can* send address selection commands to store an SC command, with the current address setting as parameter, as macro zero. The SC command enables communication, and storing it as macro zero causes it to be run upon power-up or reset.

send MD0,SC0 <return>

if the address is set to 0.

Starting a Mercury with this macro zero stored, instant communication is achieved.

### 3.4. Communication

#### 3.4.1. Test Communication

The first command used in testing communication should be a TP command. Type in a **TP** and <return>. A message indicating the current motor position should be displayed on your terminal monitor:

```
"P: +0000000012"
```

If you get a response as shown above, the **Mercury** is ready for use. All registers are loaded with their default values. Most tasks can be accomplished using these default values, but some tasks may require value modifications.

If you do not receive such a response, there are several possible reasons:

Power Failure:

Check the power supply voltage to the controller. If none of the LEDs at the front panel is light, power may be missing. Check P/S and the line voltage.

RS-232 Cable Failure

Check the cable to your terminal. It should be connected to the TxD and RxD pins from the **Mercury**, as well as to ground. Various terminals have differing pin assignments for TxD and RxD, so it may be necessary to reverse pins 2 and 3 on the RS-232 connector.

Wrong Baud Rate

Check the baud rate on your terminal. **Mercury** is designed to operate at 9600 baud after being reset. The baud rate can be changed internally by jumper setting.

Controller not Selected:

To enable communication with the controller, an address selection command must first be sent (firmware version 6.10 and newer). Make sure the proper controller address is being used and send the appropriate address selection command to the controller [ECo2].

#### 3.4.2. Setting Motion Control Parameters

Prior to commanding the connected motor or stage to move, the servo-control parameters (PID filter settings) as well as the velocity and acceleration values have to be set. If you are using host software with valid configuration files or the Mercury is set up to default to correct parameter settings, then you may skip this section.

Each stage model requires its own set of parameters for proper operation. The recommended values for the PID settings given in the following table is a good set to start with. They can be modified as appropriate for the actual setup. The settings are not particularly critical, so you can vary parameters by +/- 50% to achieve better settling performance.



### 3.4.3. Operating Motors and Stages

Type MN<return>. The red LED turns off and lights green. **Mercury** is now on-line and ready for motion.

*Note: If the motor oscillates or begins to "run away" when the MN command is given, remove power and read further before continuing.*

Enter MR1000 <return>. The motor should move 1000 encoder counts in the positive direction. When it stops, enter TP <return>. The position should be very close to 1000. Now enter TT,TE <return> in order to read the target and the position error. This reports where the motor should be (1000) and the distance of the actual position from this target. If the reported position was 998, then the reported error will be 2. If the reported position was 1002, the reported error will be -2.

To read all values at the same time, enter TT,TP,TE <return>. Now all three values are reported.

Now enter MR-1000 <return>. The motor should return to zero, or home, position. Press <return> again. The motor should move another 1000 counts in the negative direction. Press it again and again. The motor should move each time.

Now try GH <return>. The motor should return to the position it occupied when power was applied.

Now for some really advanced programming: enter MR1000,WS100,TE <return>. The motor should move 1000 counts positive. When it arrives at this position, it should report how close it came to the target.

For a longer move, enter MR100000<return> and verify that the motor has moved that distance. Press the <return> key again and see that the motor moves the same amount in the same direction. Press it again and again. The motion should continue each time the <return> key is pressed.

A GH<return> command will return to the previously defined HOME position. While the motor is in motion, give the command TE,TP <return>. You will see the distance to the target scrolling down your screen as it is approached. Since we are moving to the HOME position, we would have had the same effect if we had executed TP,WA100,RP <return>.

Or, while in motion, we could have entered TV400 and pressed the <return> key as often as desired to read the actual number of encoder counts moved during the last 0.4 seconds.

Set the acceleration to a lower value, such as SA10000 <return> and see the velocity increase and decrease over a longer period. Set the velocity to various values and issue commands to move to various positions using either MA or MR commands.

Move the stage some distance away from the end, say 100,000 counts, and try this: MR50000,WS100,WA500,MR-50000,WS100,WA500,RP9 <return>.

The stage should cycle back and forth 50,000 counts with a delay of a half second at each end. It should do this ten times—once by command followed by nine repetitions.

## 4. Command Types

Over 50 commands are available for programming the **Mercury**. Use these commands to control motion and to acquire reports regarding motion control parameters and status.

Wherever possible, the command structure has been designed to minimize the effort required to use the **Mercury**. As Everett Long used to say, "Make it as easy as using a screwdriver."

*Commands can be executed in various ways:*

|                             |  |
|-----------------------------|--|
| <b>as Single Commands</b>   | One function, executed immediately                   |
| <b>as Compound Commands</b> | Multiple functions, executed immediately in sequence |
| <b>as Macro Commands</b>    | Command sequences, stored for later execution        |

### 4.1. Single Commands

A **single command** is executed immediately after a carriage return is received and will be repeated each time a carriage return is received, until a different command is entered. With this feature, it is very easy to continuously monitor the state of an input by simply holding down the "Enter" key.

Examples:

|                         |   |
|-------------------------|---|
| <u>MR2000</u> <return>  | Move motor 2,000 steps relative to the present target |
| <u>MN</u> <return>      | Set motor in ON state                                 |
| <u>GH</u> <return>:     | Send motor to home position (go home)                 |
| <u>TP</u> <return>      | Report (tell) position for motor                      |
| <u>MA20000</u> <return> | Send motor to absolute position 20,000                |

Both uppercase and lowercase characters are valid, and spaces are allowed.

|                     |                         |
|---------------------|-------------------------|
| <u>TP</u> <return>: | Reports "P:+0000000000" |
| <u>tp</u> <return>  | Reports "P:+0000000000" |

### 4.2. Compound Commands

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

The syntax for a compound command is:

CMD[*n*], CMD[*n*], ..., <return>

Examples:

MR2000,WS100,MR-4300 <return>  
mr5000,ws100,wa500,ma12000,ws100,wa800,tp<return>

MA3500, WS100, WA120, tp<return>

A compound command, such as in the following example, may be entered as one program line. It instructs the motor 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 that sequence 5 times.

Example: "MR1000, WS100, WA500, MR-1000, WS100, WA1000, RP5" <return>

Once this compound command is entered, it remains in the buffer until replaced by another command and can be re-executed by sending a carriage return (pressing <enter>).

**Note:** *Compound commands may contain up to 19 single commands.*

### 4.3. 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, stored under a macro number. To use macros to programming the **Mercury** controller, insert an MDn (Macro Definition) command as the *first* instruction on the line and follow it by a comma and a comma-separated command string. The syntax for macro commands is: MD(macro#), followed by a compound command string.

Example: MD3, MR1000, WS100, MR-1000, WS100, RP5 <return>

In this example, MD3 defines macro #3. To call up (run) this macro, just issue the command EM3. Unlike MD# commands, EM# commands may be used in compound commands and, with some restrictions, in other macro definitions.??

Up to 32 macro sequences, each holding up to 16 single commands, can be stored in the Mercury flash memory.

31 macro sequences (numbers 1 to 31) are available for general purposes. Macro sequence #0 has a special: it is called the auto-start macro and is run automatically upon power-up or reset (see next chapter).

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 sequence in one macro and define important parameters such as torque, gain, or velocity, in another macro which is called by the main macro.

A macro command can call other macros, but if the calling macro is to continue after the called macro completes, the called macro must not contain any macro calls. For example, MC1 could call MC2, but MC2 could not then call MC3 and still be able to return to complete the remainder of MC1. A macro may call as many other macro commands as desired, as long as each one called does not call another. If there is no need to return to the calling command, then macros may call macros to any extent desired.

Example: MD1, EM2, EM3, EM4, EM5, EM6

**Note:** *Each macro command may contain up to 16 single commands.  
Up to 32 macro sequences can be stored.*

#### 4.4. Macro Zero

The Macro Zero is a special macro command. If a command sequence is stored as macro 0, it will be executed automatically immediately after a system reset or power-up. This allows specification of a motion program that is automatically executed when power is applied and is the mechanism by which stand-alone operation is implemented.

Macro zero may also be used only to set up custom parameters, either before manual control is begun or before transferring control to other macros for stand-alone operation.

Macro Zero is defined by the "MD0,xxx" command, where xxx represents a comma-separated list of single commands e.g.:

MD0,MR50000,WS100,GH,WS100,RP4<return>

Macro zero can be erased by:

RZ<return>

The contents of macro zero can be read by

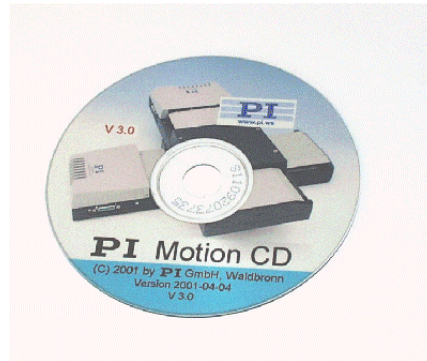
TZ<return>

#### 4.5. Reporting Commands

**Reporting commands** are commands which cause the *Mercury* controller to send a string of data, be it a position, servo-control parameter, help or other information. These commands are easy to remember as they usually begin with a **T (tell)** or **G (get)**. For example, TT (Tell Target), or GP (Get proportional gain).

## 5. Software

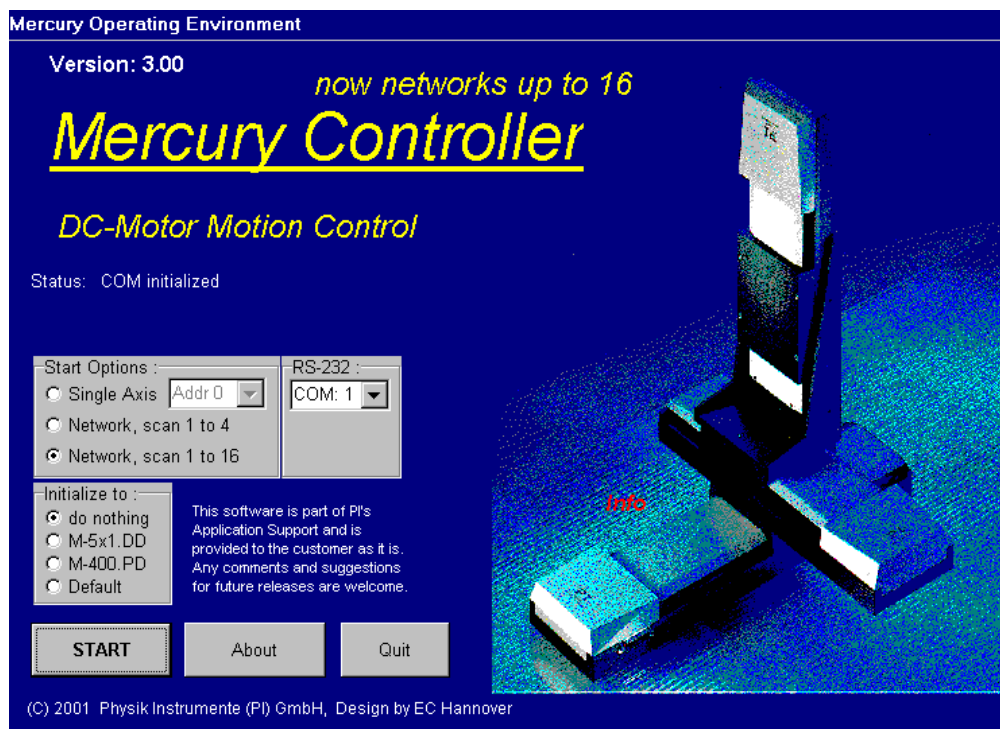
Mercury software is available on the PI-Motion CD that comes with the shipment.



### 5.1. Host Software

Mercury Controllers can be used with the Mercury Host software, a versatile operating environment for Windows 95/98/2000 and NT. This software offers a convenient alternative to sending commands to the Mercury from a simple terminal emulator.

#### Start Screen



The Mercury Software supports up to 16 Controllers connected via serial daisy-chain cables. If only one Mercury is connected, select "Single Axis" and choose the address the controller is set to.

If more than one Mercury is networked, select either "Network, scan 1 to 4" (if all addresses are smaller or equal to 4) or "Network, scan 1 to 16" (if any Mercury is set to an address larger than 4).

Note for Firmware version 6.0 or later:

Mercury controllers with firmware version 6.01 or later should not be used with Mercury Software versions older than 3.0. If you are still using version 1.31, select "Network" as start option, *even if you have only one Mercury connected.*

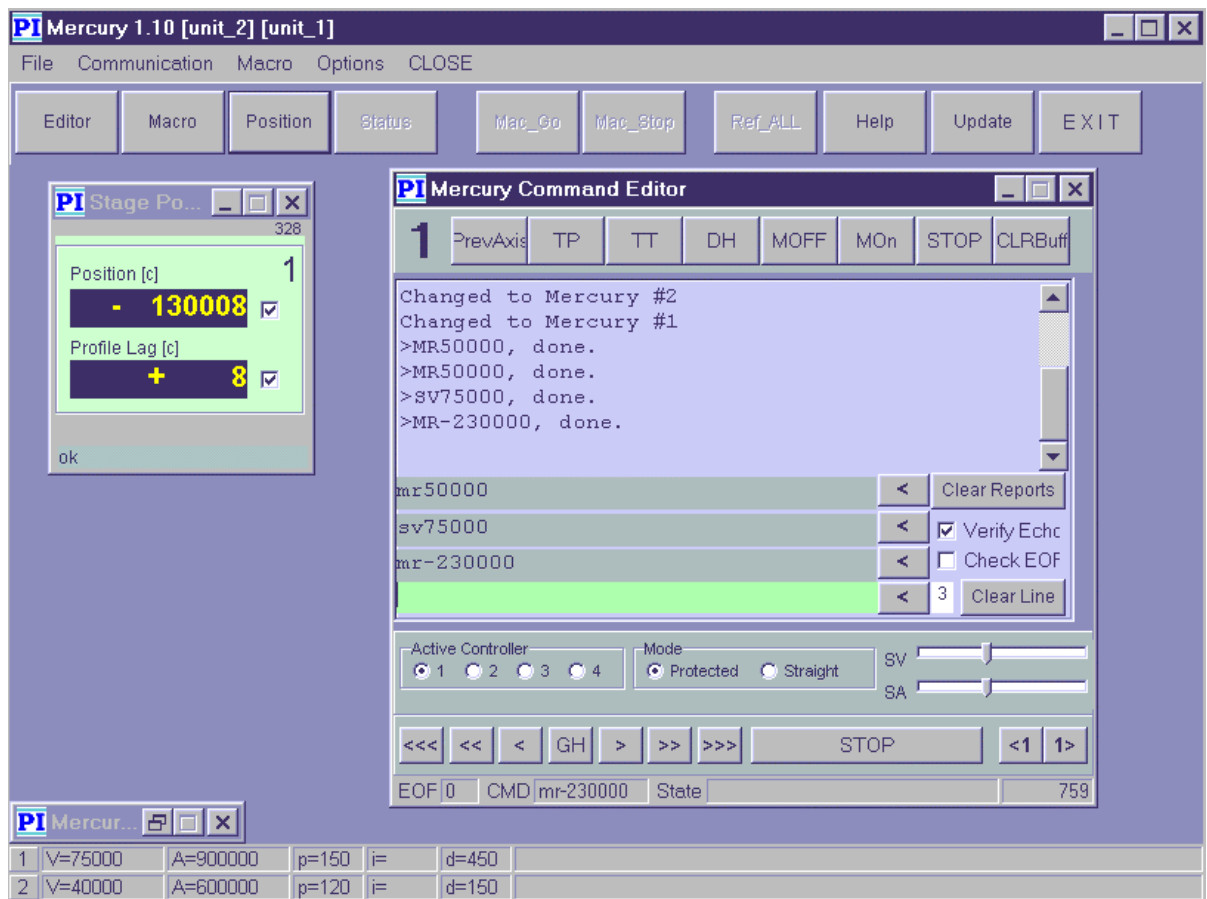
The "Single Axis" option of software versions before 3.0 does not work with firmware version 6.0 or later.

If you know that the Mercury is properly configured for the motor drive connected, then choose the "do nothing" initialize option. In this case nothing will be overwritten and no address selection command or parameter changes will be sent to the controller.

If you are using the Mercury controller the first time and you have M5x1 or M-400.DP series mechanics, choose the corresponding "initialize to" option.

Then move to the main screen with the "START" button.

### Main Screen



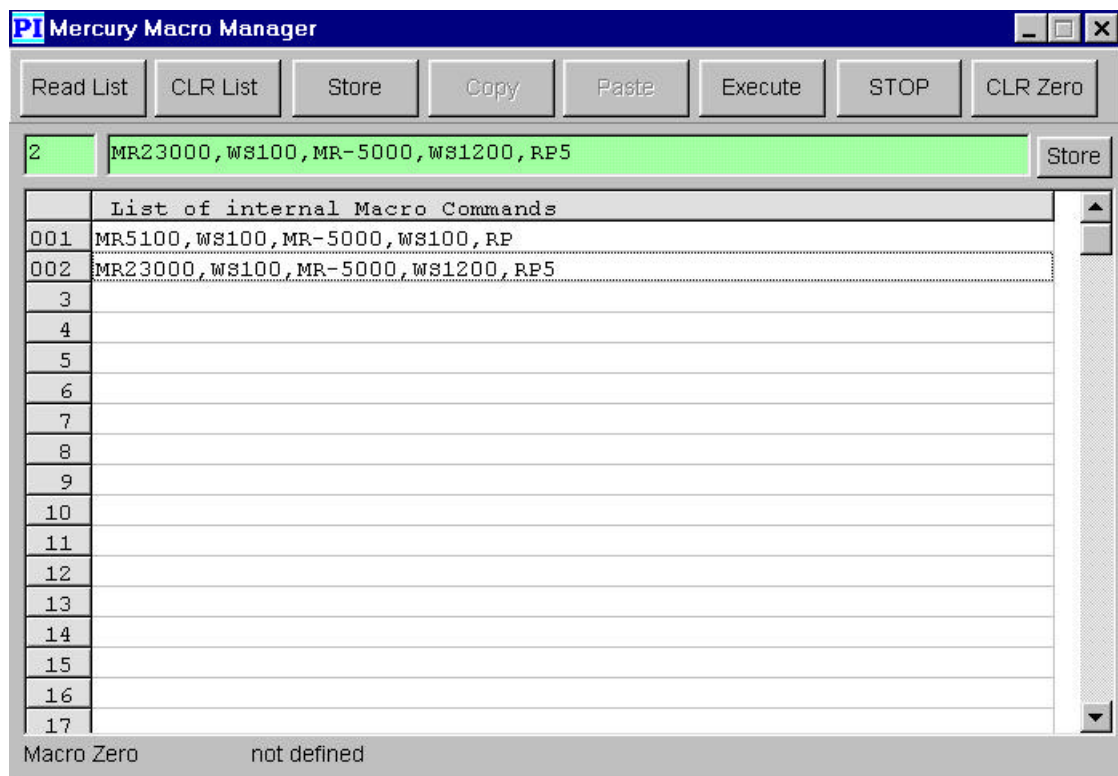
The main screen contains the Command Editor, Position Display and Macro Manager. Most of the functions are designed for intuitive operation. You may try to move the stage using the <?? command or by clicking the appropriate buttons.

Functions assigned to the buttons:

| <i>Button</i> | <i>Function</i>                       |
|---------------|---------------------------------------|
| Editor        | Opens the Command Editor?? window     |
| Macro         | Opens the Macro Manager               |
| Position      | Opens the real-time position display  |
| Help          | Shows a list of commands              |
| Update        | updates the status line entries       |
| PrevAxis      | Switches to previously active axis    |
| TP            | Displays current position active axis |
| TT            | Displays programmed target position   |
| DH            | Sets current position to zero         |
| MOFF          | Sets Motor servo off                  |
| MOOn          | Sets Motor servo on                   |
| STOP          | Motor halt                            |
| CRLBuff       | clears the communication buffer       |
| Clear Reports | clears the report window              |
| Clear Line    | clears the command line               |
| <<<           | move (-) 200,000 counts               |
| <<            | move (-) 50,000 counts                |
| <             | move (-) 10,000 counts                |
| >>>           | move (+) 200,000 counts               |
| >>            | move (+) 50,000 counts                |
| >             | move (+) 10,000 counts                |
| GH            | go to zero position                   |
| STOP          | immediate motor stop                  |
| <1            | move one step in negative direction   |
| >1            | move one step in positive direction   |

### Macro Manager

The Macro Manager offers an alternative to using the Mercury's macro management commands directly.



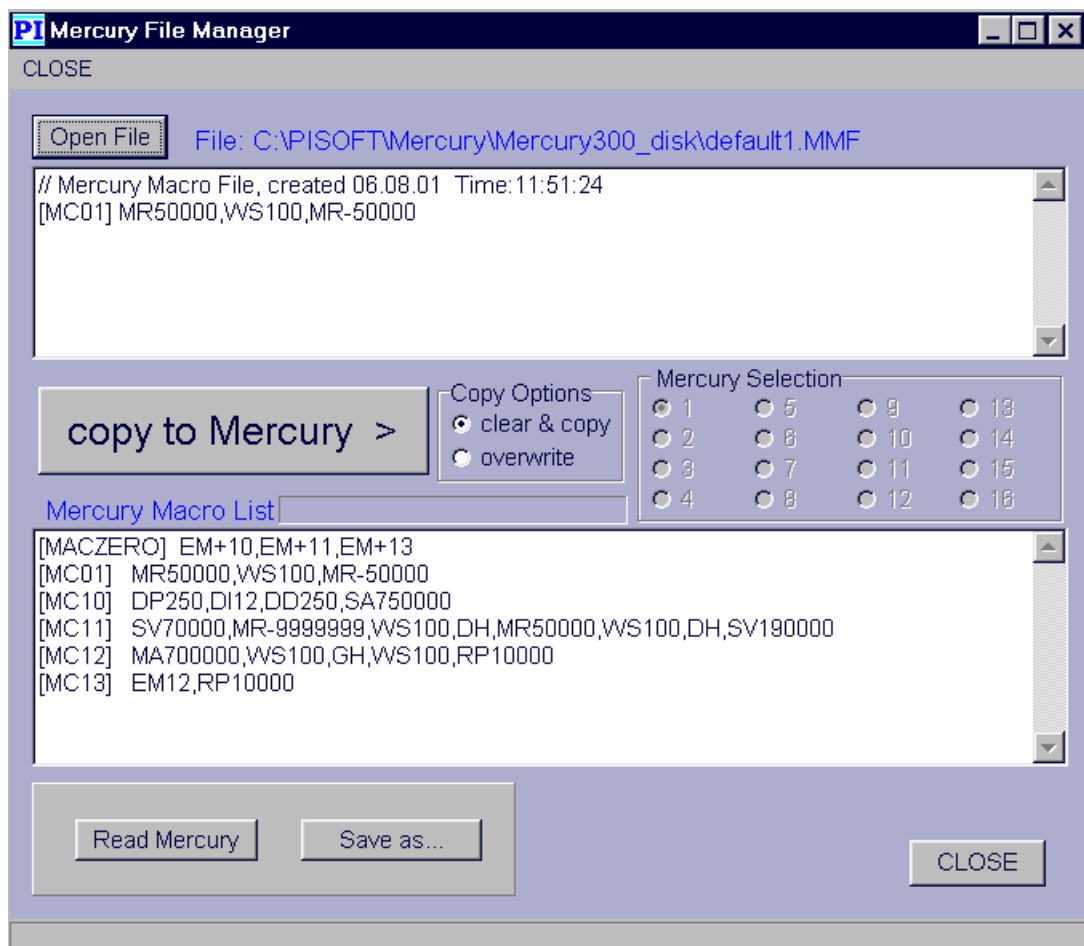
| <i>Button</i> | <i>Function</i>  |
|---------------|--|
| Read List     | Updates the macro commands in the table  |
| CLR List      | Clears the table   |
| Store         | After a new macro command is edited, stores this macro under the current number. |
| Execute       | Runs the macro shown in the edit line.   |
| STOP          | Stops a running macro  |
| CLRZero       | Clears the auto start macro (macro zero)   |

### Writing and executing macros with the Macro Manager

Click in the cell with the macro you want to edit. Then modify the macro content or create a new one and press <return> to save it permanently in the Mercury.

After the macro is saved, it can be executed by clicking on the "Execute" button.

### Macro File Manager



The Macro File Manager can be used to download/upload and copy macro sequences. The files are stored in ASCII text format, so any text editor like Notepad can be used to edit macro files.

The "clear & copy" and "overwrite" buttons affect the way the macros in the working list on the screen are copied to the Mercury when "copy to Mercury" is



pressed. If "clear & copy" is selected, any macros currently stored in the Mercury are erased before the new macros are downloaded (stored in the Mercury). When "overwrite" is selected, the Mercury macro storage is not first cleared: the only macros erased are those that are overwritten by new macros.

These buttons do not effect the operation of the "Read Mercury" or "Save As" buttons. Those operations always clear any previous contents from their destination, although "Save As" will warn the user if a file of the same name is about to be overwritten.??

## 5.2. Windows Library (DLL)

see current version of the PI Motion CD.

## 5.3. LabView Driver Library

see current version of the PI Motion CD.

# 6. Trouble Shooting

Problem (#1):           The controller communicates and reports position values (TP) , but the connected stage or motor does not move.

Possible causes:

- (a)  
*Verify that the motor brake is set to OFF (command: BF). Even some stages without a physical brake need to be in BF state for operation.*
- (b)  
*Verify that limit detection is set to the proper logical level (command: LH or LL??) and that limit detection is activated (command: LN).*
- (c)  
*If using a stage with internal PWM amplifier (.PD stages), verify that the 24VDC supply is connected to the stage.*

Problem (#2):           Both LEDs are blinking and the controller does not communicate

Possible reason:       *The firmware memory is signaling an internal error. To solve this problem, do a firmware update (or reload the same firmware version)..*

## 7. Command Reference

All **Mercury** commands use two-letters to identify the type of operation, followed by pertinent data values, if necessary. For example, *TP* by itself is adequate to display the motor position, but *MR* alone would be useless, because the system would not know how far you wished to move. All commands are checked for acceptability as they are entered. *MR*, for example, must be followed by a minimum of 1 and a maximum of 9 digits to accommodate the allowable range of motion.

**Mercury** commands are arranged by group for the following discussion.

### Motion and sequencing commands

AB Abort motion  
 DH Define home  
 GH Go home  
 MN Motor on  
 MF Motor off  
 MR Move relative  
 MA Move absolute  
 RP Repeat from beginning of line  
 WA Wait absolute time  
 WS Wait stop  
 FE Find edge (find origin position )

### Parameter setup commands

SV Set Velocity  
 SA Set Acceleration  
 DP Define p-term (proportional gain)  
 DI Define i-term (integral gain)  
 DD Define d-term (derivative gain)  
 DL Define integration limit  
 SM Set maximum following error  
 LN Limit switch operation ON  
 LF Limit switch operation OFF  
 LL Limits switches active low  
 LH Limit switches active high

### Reporting commands

### Report Identifier

|    |                          |      |
|----|--------------------------|------|
| CS | Report checksum          | C:   |
| TD | Tell dynamic target      | N:   |
| GP | Get p-term               | G:   |
| GI | Get i-term               | I:   |
| GD | Get d-term               | D:   |
| GL | Get integration limit    | M:   |
| TB | Tell board address       | B:   |
| TI | Tell iteration number    | X:   |
| TS | Tell status              | S:   |
| TM | Tell macro contents      | none |
| TZ | Tell Macro Zero          | none |
| TY | Tell programmed velocity | Y:   |

|    |                                   |      |
|----|-----------------------------------|------|
| TL | Tell programmed acceleration      | L:   |
| TE | Tell error (distance from target) | E:   |
| TP | Tell position                     | P:   |
| TT | Tell target position              | T:   |
| TF | Tell profile following error      | F:   |
| TV | Tell actual velocity              | V:   |
| VE | Display version number            | none |

#### Utility commands

|    |   |
|----|---|
| BN | Set brake ON  |
| BF | Set brake OFF   |
| EF | (Set)?? Echo OFF  |
| EN | (Set)?? Echo ON   |
| RT | Reset all parameters to default?? and do power-up start |
| UD | Update flash memory[EC03]                               |

#### Macro commands

|    |                       |
|----|-----------------------|
| MD | Macro definition      |
| EM | Execute Macro         |
| RM | Reset (erase) macro   |
| TM | Report macro contents |
| TZ | Tell Macro Zero       |

## 7.1. MOTION and SEQUENCING COMMANDS

**AB** Abort motion

This command stops the motor at the present position; any further motion requirements are ignored. The target position is changed to be equal to the present position. AB motion is used for stopping an undesired motion as well as to hold at the current position.

Example: AB<return> : Aborts motion of motor

**BN** Brake ON

Activates the motor brake, if the stage is equipped with the optional motor brake option. Actually, the voltage at pin #1 of the motor connector is set to 0V, allowing the spring-loaded brake to clamp.

With M-5x1.DD stages, the low signal at pin #1 will disable the motor amplifier even if no brake is installed.

**BF** Brake OFF

Releases the motor brake, if the stage is equipped with optional motor brake. Actually, the voltage at pin #1 of the motor connector is set to +5V, so the spring-loaded brake is cocked open.

With M-5x1.DD stages, the TTL high signal at pin #1 will enable the motor amplifier.

**FE<sub>n</sub>** Find Edge (n = 0,1)

This command is used to move the system to a given initial position. The direction of motion depends on the value of n. The target is set to + or - 1,000,000,000 (i.e. off scale), depending on the value of **n**. Then the motor is run at the programmed speed until the reference point is reached (i.e. until a transition occurs on the reference input line)..

Example: FE0 <return> causes motor to move in a positive direction until the reference input changes state. If the reference input is high when the command is issued, the motor runs toward the positive limit until the input changes to low, and vice versa.

FE1 <return> causes the motor to move in a negative direction until the reference input changes state.

These commands assume that your system has the reference input line connected to a reference switch. Otherwise, motion will continue until the target position of + or - 1,000,000,000 is attained, or until it is stopped manually or by command.

FE can be useful in calibrating the home position on start-up or for simply slewing long distances at the most recent velocity setting.

If limit switches are not in use, a command string such as "FE0,WS100,WA500,DH,MN" could be used on startup to find the Home position value of a mechanical stop. This command tells the motor to run toward a target

value of +1,000,000,000 until stopped. When the motor encounters the mechanical stop, the following error will rise quickly, triggering a halt because of an Excessive Error condition. (See SM command.) This error halt will disable the motor loop, so the MN command at the end is required to maintain the new position against any forces that might be attempting to move it. It may be desirable to use the SM command to set the maximum allowable following error to a low value at the start of motion. Do not forget to restore it to a better working value after the operation is complete.

Note: This procedure requires that limit switches not be in use. If they are in use, they will automatically stop motion without causing an Excessive Error condition.

|           |                |
|-----------|----------------|
| <b>GH</b> | <b>Go Home</b> |
|-----------|----------------|

The Go Home command causes the motor to move to the absolute zero position. Equivalent to an MA0 (Move to zero position) command.

Example:

GH <return>: Moves motor to zero position.

|                         |                      |  |
|-------------------------|----------------------|--|
| <b>MA<math>n</math></b> | <b>Move Absolute</b> | <b>(-1,073,741,824 &lt; <math>n</math> &lt; 1,073,741,823)</b> |
|-------------------------|----------------------|--|

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

MA0 <return>: Tells motor to go to Home position

|                         |                      |
|-------------------------|----------------------|
| <b>MR<math>n</math></b> | <b>Move Relative</b> |
|-------------------------|----------------------|

This command generates a motion of relative distance of  $n$  counts in the specified direction from the current motor position.  $n$  may be either a positive or negative number. The resulting absolute target position must be between + and - 1,073,741,823.

Examples:

MR5000 <return>: Motor moves 5,000 counts in positive direction.

MR-330 <return>: Motor moves 330 counts in negative direction.

MR2000,WS100,MR-1200 <return>: ??

|           |                  |
|-----------|------------------|
| <b>MF</b> | <b>Motor OFF</b> |
|-----------|------------------|

When this command is issued, the motor is no longer held in position (servo-control is turned off) and may be moved freely. 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 monitored in the MF status and may be reported by the TP command.

The opposite command is MN (Motor ON).

Use caution when turning the motor back on. The target position remains the same as when the MF command was issued and the motor will try to return there unless the target position is redefined.

Examples: MF <return> : Sets motor servo-control to OFF

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

AB,MN <return>

DH,MN <return>

The DH command is also useful for determining the encoder resolution. Issue the MF command, manually position the motor/encoder at some known angle, issue a DH command to set the position to 0, then turn the motor/encoder one revolution. Now the TP command will report the number of encoder counts/ revolution. **P I** stages have incremental encoders with from 2000 to 4000 counts/ rev.

|           |                 |
|-----------|-----------------|
| <b>MN</b> | <b>Motor ON</b> |
|-----------|-----------------|

This is the normal system control mode, in which **Mercury** controls the axis position continuously. Any deviation between actual and target position causes the motor to be driven toward the target, possibly with maximum force, depending on the distance moved during the motor off condition.

Use caution when turning the motor back on. **Mercury** keeps track of both the motor and target position even when in the MF state. When it receives an MN command, The controller will try to return the motor to the old target, unless the target has been redefined.

Example:

MN <return>

|            |               |                               |
|------------|---------------|-------------------------------|
| <b>RPn</b> | <b>Repeat</b> | <b>(0 &lt; n &lt; 65,535)</b> |
|------------|---------------|-------------------------------|

This command causes the command string to repeat *n* times. If *n* is not specified, the command(s) in the string are repeated 65,536 times. The repeat loop may be interrupted by sending any character. This character should not be the first character of a new command, because it will be discarded. (To repeat forever, use two RP commands in sequence??.)

Example: TE,WA500,RP99 <return>

Will display the distance to the target every 500 milliseconds (0.5 second) for a total of 100 times.

|            |             |                               |
|------------|-------------|-------------------------------|
| <b>WAn</b> | <b>Wait</b> | <b>(0 &lt; n &lt; 65,535)</b> |
|------------|-------------|-------------------------------|

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

Example: MR2000,WA3000,MR-2000 <return>

This command line will move the motor by 2,000 steps, then, 3 seconds after the start of the move, the motor will move back 2,000 steps. Note that the wait period of 3 seconds includes the time the motor is moving. If the motor is to be at rest for 3 seconds, an additional command must be inserted to prevent the wait interval from beginning until the motor has completed it's motion:

MR2000,WS100,WA3000,MR-2000 <return>

|            |                            |
|------------|----------------------------|
| <b>WSn</b> | <b>Wait for motor stop</b> |
|------------|----------------------------|

The WS command waits until the motor has reached the end of its trajectory and then waits for another n milliseconds before continuing to the next command. If the parameter n is omitted, the default wait time of 1 second is used.

Example: MR5000,WS100,TE <return>

This command line moves the motor for 5000 counts and then waits for 100 ms after the motor has completed the move before executing the TE command (Tell error) and reporting the error.

Example: MR5000,WS,TE <return>

This command moves the motor for 5000 counts and waits for 1 second (default value, used if no number is specified) after the motor has completed the move before executing the TE command (Tell error) and reporting the error..

## 7.2. PARAMETER SETUP COMMANDS

### DH Define Home

Defines the current motor position as the zero position (Home position).

Example: DH <return>: Sets the current motor position to 0.

### DPn Define Proportional gain (0 < n < 32,767)

This command sets the slope of the proportional relationship between the position error and the motor voltage. The higher the gain value set, the greater the stiffness of the position coupling, meaning that a small error value causes a proportionally larger motor current driving the motor towards the target.

The default gain value usually ensures stable operation. The optimum value depends on friction, inertia, motor power, and the resolution of the encoder. It must be determined by the user.

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 within acceptable limits. If the axis becomes unstable and begins to oscillate, the gain must be reduced until the oscillation stops.

Example: DP80 <return> Sets proportional gain of 80

### DI n Define Integral gain (0 < n < 32,767)

Sets the gain to be applied to the integral term in the PID algorithm. The primary function of this term is to overcome friction-induced static errors.

### DDn Define Derivative gain (0 < n < 32,767)

Sets the gain to be applied to the derivative term in the PID algorithm. The primary purpose of this term is to increase damping and reduce overshoot at the end of motion.

|     |                       |                  |
|-----|-----------------------|------------------|
| DLn | Define Integral Limit | (0 < n < 32,767) |
|-----|-----------------------|------------------|

Limits the amount of contribution by the integral gain function.

|      |                  |                         |
|------|------------------|-------------------------|
| SA n | Set Acceleration | (0 < n < 1,073,741,823) |
|------|------------------|-------------------------|

Sets the maximum?? acceleration rate in encoder counts per second squared.

Typical acceleration values are in the range from 100.000 to 800.000. The default value is set to 150.000.

|     |              |                   |
|-----|--------------|-------------------|
| SVn | Set Velocity | (0 < n < 500,000) |
|-----|--------------|-------------------|

Causes the motor to run at maximum?? velocity of n counts/s for subsequent motion commands. 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.

Example:

SV40000 <return> Sets the velocity of motion to 40,000 counts per second.

|     |                             |                  |
|-----|-----------------------------|------------------|
| SMn | Set Maximum following error | (0 < n < 32,767) |
|-----|-----------------------------|------------------|

Sets the maximum allowable error between the dynamic target and the actual position. May be changed as often as desired to provide maximum protection to the system. The normal following error can be monitored during motion with the TF command. For maximum system safety, use the SM command to limit following error to a value slightly above that required for normal operation.

|    |                             |
|----|-----------------------------|
| LN | ??Limit Switch operation ON |
|----|-----------------------------|

Enable software limit switch operation. When a limit switch is encountered during motion, motion is halted and is no longer possible in that direction as long as the switch remains closed. The target is changed to the position at which the limit switch was encountered. Movement in the reverse direction is not affected.

|    |                              |
|----|------------------------------|
| LF | ??Limit switch operation OFF |
|----|------------------------------|

Disables software limit switch operation. **Mercury** also has logic circuitry to disable motion in the direction of a hardware limit switch when the switch is actuated, whether or not the software limit switch operation is enabled. The LN and LF commands affect only the software. The LF command should only be used when hardware limit switches are not installed.

|    |                         |
|----|-------------------------|
| LL | Limit switch active LOW |
|----|-------------------------|

Sets the active state of both limit switch inputs to low. When this input is less than 1 volt and limits are enabled, motion in the corresponding direction will be terminated.

|    |                          |
|----|--------------------------|
| LH | Limit switch active HIGH |
|----|--------------------------|



Sets the active state of both limit switch inputs to high. When this input is greater than 3 volts and limits are enabled, motion in the corresponding direction will be terminated.

### 7.3. REPORTING COMMANDS

**Reporting commands** cause the **Mercury** controller to emit a string of data, be it a position, target, help string or other information. These commands are easy to remember as they usually begin with a **Tell** or **Get** statement. For example, TT (Tell Target), or TP (Tell Position) or GP (Get proportional term).

Report commands are structured to display a fixed number of digits.

Examples:

transmit: TT <return>,      receive: "T:+0000000000"

transmit: GP <return>,      receive: "G:+0000000120"

|    |                |
|----|----------------|
| VE | Version report |
|----|----------------|

Reports the copyright notice and revision level of the installed firmware.

transmit: VE <return>

Receive:

"(C)2000,1 DIVA Automation, Inc./PI GmbH Waldbronn, Ver.  
6.10, 14 July 2001"??

|    |            |
|----|------------|
| TE | Tell Error |
|----|------------|

Reports the position error of the motor, as determined by subtracting the actual position from the target position. This command also works while the motor is moving. It can thus be used to determine if the motor is actually moving, if it is moving in the proper direction, and if it is approaching the target or maintaining position without oscillation.

Example: TE,WA150,RP <return>

This example will track the calculated positional error of the motor during and after the move in progress. Every 150 ms the actual distance from the target is reported. The output may be stopped by entering any character.

Report: "E:+0000000000"

|    |                       |
|----|-----------------------|
| GP | Get Proportional term |
|----|-----------------------|

Reports the current Proportional Gain value. This value can be changed with the DP (Define Proportional Gain) command.

Report: "G:+0000000080"

|    |                   |
|----|-------------------|
| GI | Get Integral term |
|----|-------------------|

Reports the current Integral Gain value. This value can be changed with the DI (Define Integral Gain) command.

Report: "I:+0000000080"

#### GD Get Derivative term

Reports the current Derivative Gain value. This value can be changed with the DD (Define Derivative Gain) command.

Report: "D:+0000000070"

#### GL Get Integration Limit

Reports the current Integration Limit value. This value can be changed with the DL (Define Integration Limit) command.

Report: "M:+0000002000"

#### TI 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: MR100,WS100,WA250,TI,RP99 The motor will make repetitive moves of 100 steps, with a delay of 0[EC04].25 seconds between steps, for a total of 100 times. The TI command will report the number of iterations remaining to be performed after each iteration.

Report1: "X:+0000000000" (First TI is executed before the number of loops is specified!)

Report2 "X:+0000000099" and so on.

#### TP Tell Position

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

Example: TP,WA100,RP <return>

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

Report: "P:+0000005555"

"P:+0000005555"

"P:+0000005555" ...(until stopped, or 65536 times)

#### TS Tell Status

The TS command reports the status of the system and the motion and limit switches states.

The format is "S:1F 2F 3F 4F 5F 6F"

First hex digit holds bits 7 through 4??4-7, last hex digit holds bits 3 through 0??0-3.

|    |                          |   |
|----|--------------------------|---|
| 1F | LM629 status             | Bit 0 : Busy<br>Bit 1 : Command error<br>Bit 2 : Trajectory complete<br>Bit 3 : Index pulse received<br>Bit 4 : Position limit exceeded<br>Bit 5 : Excessive position error<br>Bit 6 : Breakpoint reached<br>Bit 7 : Motor loop OFF   |
| 2F | Internal operation flags | Bit 0 : Echo ON<br>Bit 1 : Wait in progress<br>Bit 2 : Command error<br>Bit 3 : Leading zero suppression active<br>Bit 4 : Macro command called<br>Bit 5 : Leading zero suppression disabled<br>Bit 6 : Number mode in effect<br>Bit 7 : Board addressed  |
| 3F | Motor loop flags         | Bit 0 :<br>Bit 1 :<br>Bit 2 : Move direction ??polarity<br>Bit 3 : Move error (MF condition occurred in WS)<br>Bit 4 :<br>Bit 5 :<br>Bit 6 : Move error (Excess following error in WS)<br>Bit 7 : Internal LM629 communication in progress  |
| 4F | Limit Switch status      | Bit 0 : Limit Switch ON<br>Bit 1 : Limit switch active state HIGH<br>Bit 2 : Find edge operation in progress<br>Bit 3 : Brake ON<br>Bit 4 :<br>Bit 5 :<br>Bit 6 :<br>Bit 7 :  |
| 5F | Limit switch inputs      | Bit 0 :<br>Bit 1 : Reference switch input<br>Bit 2 : Positive limit switch input<br>Bit 3 : Negative limit switch input<br>Bit 4 :<br>Bit 5 :<br>Bit 6 :<br>Bit 7 :   |
| 6F | Error Codes              | (numbers reported):<br>01 : command not found<br>02 : First command character must be a letter<br>05 : Character following command must be a number<br>06 : Value too large<br>07 : Value too small<br>08 : Continuation character must be a comma<br>09 : Command buffer overflow<br>0A : macro storage overflow |

The data is presented in hex form. For those not familiar with hexadecimal, it will require some practice to make use of this command.

|           |                    |
|-----------|--------------------|
| <b>TT</b> | <b>Tell Target</b> |
|-----------|--------------------|

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.

Command: TT <return>

Report: "T: +0000000000"

|           |                                 |
|-----------|---------------------------------|
| <b>TY</b> | <b>Tell programmed velocity</b> |
|-----------|---------------------------------|

Reports the Programmed Velocity value. This value can be changed with the SV command. The values reported with the TV and TY commands should differ by only a few counts.

Command: TY <return>

Report: "Y: +0000020000"

|                         |                             |  |
|-------------------------|-----------------------------|--|
| <b>TV<math>n</math></b> | <b>Tell actual Velocity</b> | <b>(1 &lt; <math>n</math> &lt; 65,535)</b> |
|-------------------------|-----------------------------|--|

Measures and reports the number of encoder counts moved during the previous  $n$  milliseconds.

If the parameter is omitted, the time period used is 1000 ms (1s).

Command: TV300 <return>

Report: "V: +0000020006"

|                           |                    |   |
|---------------------------|--------------------|---|
| <b>TM[<math>n</math>]</b> | <b>Tell Macros</b> | <b>(0 &lt; <math>n</math> &lt; 127)</b> |
|---------------------------|--------------------|---|

Displays all currently stored macro commands. If  $n = 0$  or, if  $n$  is not specified, all macros will be displayed. Since macros may be defined in any order, the TM command is useful for confirming the existence of, as well as listing its contents.

Report: "MC001\_ xxx, xxx, xxxCR  
ETXMC002\_ xxx, xxxCR  
ETXMC003\_ xxx, xxxCR  
ETXETX"

where "\_"=space, CR=carriage return, ETX=end of text character

see also: Reset Macro (RM)

|           |                        |
|-----------|------------------------|
| <b>TZ</b> | <b>Tell macro Zero</b> |
|-----------|------------------------|

TZ reads Macro Zero.

Macro Zero is defined by the "MD0,xxx" command and is automatically executed after power-on or reset or .

Report: "MC000\_ xxx, xxx, xxxCR  
ETXETX"

See also: "Macro Zero" and the RZ (reset macro zero) command

**TB      Tell Board address (Mercury address setting)**

Reports address of the currently enabled Mercury controller. The address is set by jumpers, see the "jumper setting" section. The default setting is 0

Report:        "B:0000"

If the next Mercury to be selected is set to address #1, its response to a TB command would be "B:0001".

**CS      CheckSum**

Calculates and reports the check-sum over the entire contents of the processor ROM?? and the EEPROM. Both values are reported by the CS command.

This command is useful to test the integrity of the firmware. The reported value should be the same every time.

Report:        "C:50EE\_FBF2"CRETX

**TL      Tell acceLeration**

Reports programmed acceleration value.

Command: TL<return>

Report:        "L:+0000170000"

**TF      Tell Following error**

Reports the difference between the dynamic target and the actual position. During motion, it is normal for the actual position to lag behind the target position by some amount, usually dependent on the programmed velocity. If the velocity is higher than physically possible for the system, or if an obstruction has been encountered, the Following Error will increase. If the obstruction is temporary, the servo-action will attempt to reduce the error to zero when the obstruction is removed. If the condition is not temporary, the error will typically increase until the programmed limit is reached.

Command: TF<return>

Report:        "F:+0000000117"

**TD      Tell Dynamic target**

Reports the instantaneous value of the dynamic target. As the motor is moved along the programmed path to the (final) target, a "dynamic target" is used to define the trajectory and control the position at each instant along the way to.

Command: TD<return>

Report:        "N:+0000126317"

## 7.4. MACRO COMMANDS

**EM $n$     Execute Macro Command  $n$     ( $1 < n < 127$ )**

This command is used to run a previously defined macro command. If there is no macro defined for the number  $n$ , no action will be taken.

Example: EM3 <return>                      Execute macro #3

Before calling EM3, that macro should have been defined with the MD3 command.

**MD $n$     Macro Definition                      ( $1 < n < 127$ )**

This command is used to define a new macro command. Defining more than one macro with the same value of  $n$ , will simply result in the loss of all but the last macro so defined. To define a macro, choose the desired macro number in the allowable range, enter **MD** followed by this number and a comma, and then type the command or command sequence just as you would if running it directly[EC05].

Examples:

MD1,MR50000,WS100,GH <return>                      Defines macro #1

MD2,TT,TP <return>    Creates a macro command to Tell Target, then Tell Position and assigns it to number 2.

EM2 <return>                      Executes macro #2, (Same as entering TT,TP<return>)

**RM    Reset Macro**

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

Example: RM <return>                      Removes all stored macros

Note: This command requires about 4 to 5 seconds for execution. During that time no communication is possible[EC06].

## 7.5. UTILITY COMMANDS

**RT    Reset**

Restarts the internal firmware operation, as if from a power-off condition. All values are restored to their defaults. If the non-volatile RAM option is installed, stored macros are retained. If Macro 0 exists, it will be executed[EC07].

**EN    Echo on**

Enables echoing of command characters as they are entered. Each character received is echoed unchanged. This is a very useful feature when the Suprmtr6 is being controlled manually from a terminal.

|    |          |
|----|----------|
| EF | Echo off |
|----|----------|

Disables echoing. When control is from a computer program, it is sometimes easier to program if there is no echo, unless the program uses it for verification of successful transmission.

|    |                     |
|----|---------------------|
| UD | Update Flash Memory |
|----|---------------------|

This command stores the currently set parameters into the non-volatile flash memory (as the new defaults). When the Mercury is powered up the next time, it is these parameters that will be active.

