



MS 74E User Manual

**Mercury
C-862**

**Networkable Single-Axis
DC-Motor Controller**

R 8.41

Product Description and Operating Notes



This document is valid for the following product:

C-862.xx Mercury DC-Motor Controller

Release: 8.41
Release Date: 2004-07-05

© **PI** (Physik Instrumente) GmbH & Co. KG
76228 Karlsruhe, Germany FAX: (+49)721-4846-299

E**M**ail:info@pi.ws

www.pi.ws

Table of Contents:

0	Manufacturer Declarations	4
0.1	Declaration of Conformity.....	4
0.2	Warnings and Safety Instructions	4
0.3	Quality and Warranty Clauses	5
1	Introduction	6
1.1	Product Survey and Contents of Delivery	7
1.2	Front and Rear Panel Elements.....	8
2	Quick Start	8
3	System Description	9
3.1	Firmware Version Notes.....	9
3.2	Control of Multiple Axes	9
3.3	LED Status Indicators	10
3.4	Default Values.....	10
3.5	Cable Connections.....	11
3.6	Limit Sensors	11
3.7	Position Reference Sensors	12
3.8	DIP Switch Settings.....	13
3.8.1	Address (SW1-SW4) and Baud Rate Setting (SW5-SW6).....	13
3.9	Internal Jumper Settings	14
3.10	Firmware Update.....	14
3.11	Input / Output Lines.....	15
4	Operating Notes	16
4.1	Test Communication	16
4.2	Setting Motion Control Parameters.....	16
4.3	Operating Motors and Stages.....	16
4.4	Networking	17
4.5	Address Selection Code.....	18
5	Command Types	19
5.1	Base Commands.....	19
5.2	Single-Character Commands.....	19
5.3	Compound Commands.....	20
5.4	Macro Commands.....	20
5.5	Reporting Commands	20
6	Working with Macro commands	21
6.1	Basic Macro Operation	21

6.1.1	Defining a Macro.....	21
6.1.2	Running a Macro.....	21
6.1.3	Stopping a Macro.....	21
6.1.4	Limitations.....	21
6.1.5	Macro #0 (autostart macro).....	22
6.2	Stand-Alone Operation.....	22
6.2.1	Pushbutton.....	22
6.2.2	Macros under Pushbutton Control.....	23
7	Software	28
7.1	Host Software.....	28
7.1.1	Start Screen.....	29
7.1.2	Main Screen.....	30
7.1.3	Position Window.....	32
7.1.4	Macro Manager.....	32
7.1.5	Program Manager.....	34
7.1.6	Mercury File Manager.....	32
8	Mercury Joystick Control	34
9	Troubleshooting	35
10	Command Reference	36
10.1	Command and Response Formats.....	36
10.1.1	Command execution.....	36
10.1.2	Command codes.....	36
10.1.3	Echo, Errors and Reports.....	36
10.2	Command Survey (Firmware Version 8.40).....	37
10.2.1	Address selection codes.....	37
10.2.2	Commands in Alphabetic Order.....	38
10.2.3	Motion and sequencing commands.....	39
10.2.4	Parameter setup commands.....	39
10.2.5	Report commands.....	40
10.2.6	Utility commands.....	40
10.2.7	Macro commands.....	40
10.2.8	I/O commands.....	41
10.2.9	Single-Character Commands (Firmware 8.40).....	41
10.3	Command Reference in Alphabetic Order.....	42
11	Appendix	57
11.1	Dimensions.....	57
11.2	Motor-Connector Pin Assignment.....	58
11.3	I/O Connector.....	59

0 Manufacturer Declarations

0.1 Declaration of Conformity

The manufacturer,

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



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 control signals capable of driving high-power stages and 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!

0.3 Quality and Warranty Clauses

Certification

PI (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 **PI** 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 unauthorised modification or misuse, to operation outside of the environmental specifications for the product, or to improper site preparation or site maintenance.

The design and connection of any circuitry to this product is the sole responsibility of the Buyer. PI does not warrant the Buyer's circuitry or malfunctions of PI products that result from the Buyer's circuitry. In addition, PI does not warrant any damage that occurs as a result of the Buyer's circuit or any defects that result from Buyer supplied products.

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

Disclaimer of Responsibility

PI 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. **PI** 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 a complete stand-alone control system for the smaller motors typically used in high-precision positioning systems in a very compact, single package.

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.



Fig. 1:C-862.10 Mercury Controller, dollar and euro coins for size comparison

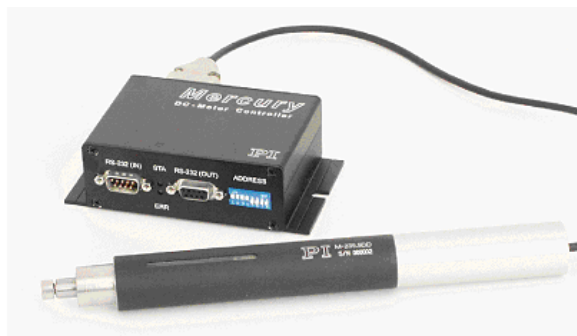
1.1 Product Survey and Contents of Delivery

Product	Contents of Delivery
C-862.00	Mercury Controller Set , including Mercury Controller (order# C-862.10) Power Supply (order# C-890.PS) Power Supply line cables (german and US types) (*) RS-232 null modem cable for PC connection (order# C-815.34) (*) PI Motion CD (software package) (*) Network cable (order# C-862.CN) (*) MS 74E User Manual (this document) (*) <i>Note: These parts are included in the C-862.10 sub-set</i>
C-862.10	Mercury Controller Unit , including Mercury Controller (order# C-862.10) RS-232 null modem cable for PC connection (order# C-815.34) Network cable (order# C-862.CN) Motion CD (software package) MS 74E User Manual (this document)

Accessories (ordered separately)

C-815.34	RS-232 Null Modem Cable
C-815.38	Stage/Motor cable, 3m (Dsub 15 m/f)
C-862.CN	Network Cable for Mercury-to-Mercury daisy-chain connection, length 28 cm
C-862.CN2	Network Cable for Mercury-to-Mercury daisy-chain connection, length 180 cm
C-862.IO	Input/output cable for digital IO connector, length 1m
C-862.PB3	Push button pad
C-890.PS	Mercury Power Supply, 15 VDC
C-890.P12	Mercury Power Supply, 12 VDC
C-990.CD	Pi Motion CD including Mercury Software Tools

All Mercury controllers are shipped with a Windows™ software application package. A power supply (C-890.PS) is included only in the C-862.00 set.



Dwg: Merc+M230.tif

*Fig. 2: One of many typical Applications:
Mercury Controller with a M-235.5DG Linear Actuator.*

1.2 Front and Rear Panel Elements



Fig. 3: Front and rear views of Mercury

Front			Rear		
RS-232 (in)	Sub-D 9(m)	Serial data input	12 to 15VDC	barrel connector	Power supply
RS-232 (out)	Sub-D 9(f)	Serial data output (network next Mercury)	DIGITAL I/O		digital input / output
STA	LED red/green	Motor Status: green: Servo ON red : Servo OFF	MOTOR	sub-D 15(f)	Motor /Stage connection
ERR	LED red/green	Command Error: green: Command OK red : Command error			
ADDRESS	DIP	Address select switches			

2 Quick Start

This Quick Start assumes that you are connecting one or more Mercury controllers together to a single RS-232 port on a host computer, and connecting a motorized axis to each controller.

1. Make sure each Mercury controller is set to a suitable device address. The factory default address setting is address 0 (this is device number 1). Each Mercury controller in a network has to be set to a unique address. The device address is set with the DIP switches on the front panel (see p. 8).
2. Connect an RS-232 null-modem cable (F-F) between the RS-232 IN socket of the first Mercury to the desired COM port of the host computer.
3. If there are to be additional Mercurys in the network, daisy chain them from RS-232 OUT to RS-232 IN of successive units. Use straight-through RS-232 cables (M-F).
4. Connect each motorized axis to the corresponding controller.
5. Connect power to the Mercury controllers
6. Connect separate power to any PWM motors as required
7. Copy and start the MMCRun software on the host PC
8. Before sending any commands to a Mercury controller, you must enable communication with it. This is usually done by sending an address selection code with the address of the desired unit over the network. The address selection remains valid (and need not be repeated) until another address is selected or the Mercury is powered down or reset. If you wish to have one particular device ready for commands upon power-up without address selection, you can place an SC(select controller) command in the autostart macro for that device (see p. 18)
9. The network should be ready for testing

3 System Description

3.1 Firmware Version Notes

Version 8.40 (released June 2004)

Checksum: CS=50EE E330

- ✍ Added short-pulse timers and commands CA, CB
- ✍ Single-character commands added/modified

Version 8.10 (released November 2002)

Checksum CS=50EE 9663

- ✍ Single-character commands added

Version 8.00 (released May 2002, obsolete)

Checksum CS=50EE 6D5B

- ✍ Modified stop commands: AB, AB1, ST
- ✍ New commands FE2 and FE3
- ✍ Full I/O support
- ✍ Analog input enabled
- ✍ Improved limit sensor recognition

Version 6.10 and earlier: Obsolete Firmware versions

3.2 Control of Multiple Axes

Mercury controllers can be networked.

Up to 16 Mercury Controllers can be connected to one RS-232 port in a daisy chain. The networking feature permits addressing each controller individually. During communication with a controller, all other controllers in the network will stop communication, but unconcluded motion of the connected axes, servo-control and/or macro execution continues. When desired, communication with the selected controller can be concluded and communication with a different controller initiated.

Switching between controllers requires sending of the ASCII character 0x01 followed by an address-number character. This sequence will be referred to as an "address selection code". 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 except address selection codes.

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 direct communication from controller to controller, only from controller to the PC. The host program must handle address selection and motion sequencing among different controllers. This communication model sets 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 commands that issue reports over the link. In this manner, each Mercury on the bus can be addressed, programmed to execute a 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, p. 17 for programming examples.

3.3 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 ERR LED (lower LED, signalizes command errors) 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 occurs during transmission of commands to the Mercury, the ERR LED will glow red until the next valid character is received. If a command is entered without achieving the expected results, check the ERR LED to see whether it is red or green.

Note: 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. Some commands like UD (update: busy for about 0.5 seconds) and RM (busy for about 2 seconds) cause LED brightness fluctuations.

3.4 Default Values

The 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:

Setting	Command to change	Factory Default
velocity:	SV	6000
acceleration:	SA	150000
P-term:	DP	35
I-term:	DI	0
D-term:	DD	0
I-Limit:	DL	2000
Echo mode	EF, EN	EF
Limit logic level	LH, LL	LH
Limit enable	LN, LF	LN
Brake mode	BN, BF	BN

Note 1: The 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 suitable 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.

3.5 Cable Connections

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

The host PC connects with a null modem cable to the RS-232(in) connector of the first Mercury in the system. The RS-232(out) connector can be used to connect an additional Mercury with a straight-through network cable (order number C-862.CN). Up to 16 Mercurys can be networked in this way, forming a daisy chain network. Note that individual controller addresses must be properly set for network operation.

Power (typically 15 V) must be connected to the barrel connector, with the positive lead in the center.

The sub-D-15 connector for the motor connection is compatible with all PI stages and motor drives. Use the C-815.38 *Motor Cable* for stage connection.



Fig. 4:C-862.CN: Mercury network cable, length 28 cm, other lengths on request

3.6 Limit Sensors

During operation, limit sensors (switches) can be used to stop motion at the end of the allowable travel range. Each of the two switches will interrupt motion in a particular direction. If positioning equipment from PI is used, the limit switches or sensors are prewired for operation with Mercury controllers. When connecting other motor drives or mechanics, the correct limit switch wiring must be determined. To do this, place the system in 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 controller can be configured to accept either an active-high stop signal (+5V) or an active-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.

Limit Signal inputs have 1 k-ohm pull-up to 5 V, so “no connection” results in a high state.

3.7 Position Reference Sensors

Reference signal detection can be used to determine precisely the absolute physical position of the stage. Since the incremental encoders used for position feedback measure relative motion only, and all counters are reset during power up, axis position at power-up is generally unknown. A position reference switch or sensor (origin sensor) in conjunction with a capture mechanism can be used to provide this information.

The FE, FE1, FE2 and FE3 commands are used to start a reference signal search run:

Command	Start the reference position search in
FE <CR>	In positive direction
FE1<CR>	In negative direction.
FE2<CR>	In positive direction if reference signal is positive, In negative direction if reference signal is negative.
FE3<CR>	In negative direction if reference signal is positive, in positive direction if reference signal is negative.

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

Reference signal input has 1 k-ohm pull-up to 5 V.

3.8 DIP Switch Settings

Function of the 8-bit switch labeled "ADDRESS"

- Switch 1 to 4: Device address (0 to 15)
- Switch 5 and 6: Baud rate
- Switch 7 and 8: not used



slider up : ON
slider down OFF

Fig. 5:

3.8.1 Address (SW1-SW4) and Baud Rate Setting (SW5-SW6)

	Address setting				Baud rate 9600		SW7	SW8
	SW1	SW2	SW3	SW4	SW5	SW6		
Address 0	ON	ON	ON	ON	OFF	OFF	OFF	OFF
Address 1	ON	ON	ON	OFF	OFF	OFF	OFF	OFF
Address 2	ON	ON	OFF	ON	OFF	OFF	OFF	OFF
Address 3	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF
Address 4	ON	OFF	ON	ON	OFF	OFF	OFF	OFF
Address 5	ON	OFF	ON	OFF	OFF	OFF	OFF	OFF
Address 6	ON	OFF	OFF	ON	OFF	OFF	OFF	OFF
Address 7	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
Address 8	OFF	ON	ON	ON	OFF	OFF	OFF	OFF
Address 9	OFF	ON	ON	OFF	OFF	OFF	OFF	OFF
Address 10	OFF	ON	OFF	ON	OFF	OFF	OFF	OFF
Address 11	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF
Address 12	OFF	OFF	ON	ON	OFF	OFF	OFF	OFF
Address 13	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF
Address 14	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF
Address 15	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Note: The (non-standard) baud rate of 19200 can be set with SW5 OFF and SW6 ON

Serial communication is set at 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 PC connection (order # C-815.34).

3.9 Internal Jumper Settings

There are three internal jumpers used for service and factory testing. The top cover has to be removed to get access to the jumpers on the printed circuit board.

- Jumper X3: **Watchdog timer**
Normal operation mode: Jumper Installed
 Firmware update mode: Jumper not installed
- Jumper X2: **Processor Reset**
Normal operation mode: Jumper not Installed
- Jumper X1: **Operating Mode**
Normal operation mode: Jumper at position closest to edge
 Firmware update mode: Jumper at position farthest from edge

3.10 Firmware Update



Fig. 6: Firmware update program interface

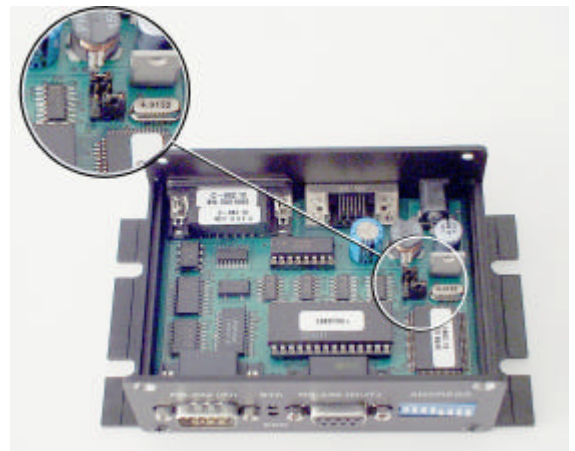


Fig. 7: internal jumpers

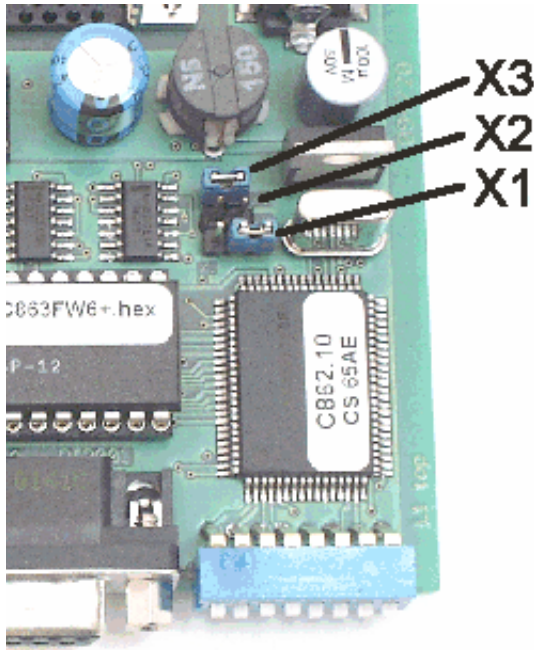
The variable part of the firmware is located in EEPROM and can be updated over the serial link.

To update to firmware version 840, for example, you need the following files:

MercUp840.EXE
 MercFW840.DAT

Firmware Update Procedure

1. Disconnect Power
2. Open the Mercury controller top cover
3. Remove jumper X3 (see chapter for jumper setting)
4. Move jumper X1 to left (inner) position
5. Connect power and RS-232 cable
6. Start the update software (*MercUp840.EXE*) and click Start Button
7. Wait about 3 minutes until data transfer has completed.
8. Disconnect power
9. Set jumpers back Install jumper X3 and set jumper X1 back to right (outer) position
10. Update is now completed.
11. Connect power again and test proper communication running the Mercury Host Software.

**3.11 Input / Output Lines**

There are 4 lines available as digital inputs and 4 lines as digital outputs (TTL). Input lines 1 to 3 also have analog input capability. The state of these lines can be read out programmatically as either digital or analog values.

A variety of commands is available to control the I/O lines, and these commands may be used with conditional statements in macros, including the autostart macro.

The following commands handle the I/O lines:

CN, CF, CP, TC, TA, WN, WF, XN, XF (for descriptions see command reference)

4 Operating Notes

4.1 Test Communication

The first command used in testing communication should be a TP command (be sure it is preceded by the necessary address selection code). Type in a **TP** and <CR>. A message indicating the current motor position should be displayed on your terminal monitor:

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

If you get a response like that shown above, the Mercury controller is working properly. All registers are loaded with their default values. Most tasks can be accomplished using these default values, but some tasks may require modified values.

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 on the front pane is lit, there is probably no power. Check the power supply and the line voltage.

RS-232 Cable Failure

Check the RS-232 cable(s).. They must carry the TxD, RxD and ground lines. Between the first Mercury controller and the host PC, a crossover (null-modem) cable is required.

Wrong Baud Rate

Check the baud rate on your terminal. Mercury controllers are designed to operate at 9600 baud, but the baud rate can be changed internally by jumper setting.

Controller not Selected:

To enable communication with a given Mercury controller, an address selection code must first be sent (firmware version 6.10 and higher). Make sure the controller address setting (DIP switches) agrees with the address in the address selection code. Remember that address selection is required even if there is only one Mercury controller connected.

4.2 Setting Motion Control Parameters

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

Each stage model requires different parameter values for proper operation. The recommended values for the PID settings given in the following table are good for initial operation. They can be fine tuned as appropriate for the particular setup. The settings are not particularly critical, so you can vary parameters by +/- 50% to achieve better performance.

4.3 Operating Motors and Stages

Type MN<CR>. The red LED turns off and lights green. The Mercury controller 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 <CR>. The motor should move 1000 encoder counts in the positive direction. When it stops, enter TP <CR>. The position should be very close to 1000. Now enter TT,TE <CR> 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 <CR>. Now all three values are reported.

Now enter MR-1000 <CR>. The motor should return to zero, or home, position. Press <CR> 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 <CR>. The motor should return to the position it occupied when power was applied.

Now for some really advanced programming: enter MR1000,WS100,TE <CR>. 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<CR> and verify that the motor has moved that distance. Press the <CR> 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 <CR> key is pressed.

A GH<CR> command will return to the previously defined HOME position. While the motor is in motion, give the command TE,TP <CR>. 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 <CR>.

Or, while in motion, we could have entered TV400 and pressed the <CR> 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 <CR> 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 <CR>.

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.

4.4 Networking

Mercury controllers can be chained together in a daisy-chain network and all controlled through one RS-232 port of the host computer. Communication is always between the host computer and one of the Mercurys, the other Mercurys will have communication disabled.

Mercury Controllers have communication disabled upon power-up or reset, although the autostart macro (if any) will be executed. The host computer can enable communication by sending an 2-character address selection code on the RS-232 link. It will usually use this mechanism to enable communication with the controller it

wishes to command, even if there is only one Mercury controller connected. Alternatively, an SC (select controller, p. 34) command in the autostart macro (p. 22) of one of the Mercurys could be used to enable communication with that device.

Even if you have only one controller connected, first send the individual address selection code (see below) to initiate communication.

The address of the controller is set by the *Address* switches. After sending the address selection code, you can verify the correct activation by sending the TB command. If there is a Mercury controller connected with the same address as the one most recently selected, it will respond to the TB command with the same address. If there is no response, then the address did not match that of any Mercury controller connected to the network.

4.5 Address Selection Code

After power-up or reset, communication is disabled. To enable communication, the address code has to be sent. When the address code does not match the address set by the address switches, the communication is disabled. In case of a Mercury network, where all Mercury controllers are set to individual addresses, sending an address code means, all controllers are disabled except that one with the matching address setting.

The enabled controller will listen for, interpret and respond to subsequent commands as appropriate. The enabled controller will stay enabled unless it is disabled by another address command.

The address selection command consists of 2 characters: the ASCII character 01 (ctrl-A) followed by an ASCII numeral 0-9 or letter A-F for addresses 0-9 and 10-15 respectively.

List of Address Codes:

<i>Device Address</i>	<i>Device Number</i>	<i>A d d r e s s C o d e</i>	
		<i>1st character</i>	<i>2nd character</i>
0	1	0x01	0x30 (decimal 48)
1	2	0x01	0x31 (decimal 49)
2	3	0x01	0x32 (decimal 50)
3	4	0x01	0x33 (decimal 51)
4	5	0x01	0x34 (decimal 52)
5	6	0x01	0x35 (decimal 53)
6	7	0x01	0x36 (decimal 54)
7	8	0x01	0x37 (decimal 55)
8	9	0x01	0x38 (decimal 56)
9	10	0x01	0x39 (decimal 57)
10(A)	11	0x01	0x41 (decimal 65)
11(B)	12	0x01	0x42 (decimal 66)
12(C)	13	0x01	0x43 (decimal 67)
13(D)	14	0x01	0x44 (decimal 68)
14(E)	15	0x01	0x45 (decimal 69)
15(F)	16	0x01	0x46 (decimal 70)

5 Command Types

Over 50 commands are available for programming the Mercury controller. Use these commands to set parameters, control motion and to read values from the controller.

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 to use as a screwdriver."

Commands can be classified in the following groups:

✎ <u>Base Commands</u>	One function, execution beginning immediately (note: the command to begin execution of a previously stored macro is a base command)
✎ <u>Single-Character Commands</u>	One character, immediate report or action
✎ <u>Compound Commands</u>	Multiple functions, executed one immediately after the other
✎ <u>Macro Commands</u>	Commands to create, modify or delete macro command sequences

5.1 Base Commands

A *base command* is executed immediately after a carriage return, <CR>, is received. The command will be repeated each time a blank line (another <CR> with nothing before it) is received. This feature makes it very easy to continuously monitor the state of an input by simply holding down the <ENTER> key.

Examples:

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

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

TP <CR>:	Reports "P:+000000000"
tp <CR>	Reports "P:+000000000"

5.2 Single-Character Commands

Single character commands are used to generate a report by sending only one character without a trailing <CR>. This allows faster requests (saving the second character and the <CR>), but the main advantage of single character commands is that they can be used during execution of macros or compound commands without interrupting a running sequence.

Find a complete overview on single character commands in the command reference.

5.3 Compound Commands

A *compound command* is a series of base 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*],, <CR>

Examples:

MR2000,WS100,MR-4300 <CR>

mr5000,ws100,wa500,ma12000,ws100,wa800,tp<CR>

MA3500, WS100,WA120,tp<CR>

The compound command in the following example 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 the sequence 5 times.

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

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 <CR>.

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

5.4 Macro Commands

Macros can be a most powerful tool for the programmer. A *macro command* is a Base Command or a Compound Command stored under a macro number.

Learn more about macro commands in the next chapter.

6 Reports

Some commands cause the Mercury controller to send a string of data to the host computer, be it a position, servo-control parameter, help or other information. These commands, sometimes called *reporting 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). There are reporting commands of all of the command types listed above.

7 Working with Macro Commands

Macros can be a most powerful tool for the programmer. A *macro command* is a Base Command or a Compound Command stored under a macro number inside the Mercury Controller.

7.1 Basic Macro Operation

7.1.1 Defining a Macro

To define a macro command, insert an MD n (Macro Definition) command as the *first* instruction in a compound command. The remaining command or commands in the compound command will be stored in the Mercury controller as macro number n .

Example 1:

```
MD1,MR5000 <CR>
```

With this command string, the base command "MR5000" is stored as macro command #1. The move command is not executed until the macro command is executed.

Example 2:

```
MD3,MR1000,WS100,MR-1000,WS100,RP5 <CR>
```

With this command, the compound command "MR1000,WS100,MR-1000,WS100,RP5" is stored as macro #3.

7.1.2 Executing (Starting) a Macro

To execute (call up, run) the macro defined above, just issue the command EM3. Unlike MD# commands, EM# commands can be used anywhere in compound commands.

7.1.3 Stopping a Macro

Macro commands terminate either naturally when they have completely executed or they can be stopped when the controller is addressed and receives any character except a single-character command, <CR>, or an address selection code.

Example 3:

Assuming the macro "MR500,WS200,RP10000" is running and will not terminate until 10000 moves of 500 counts each are completed. If you want to stop this sequence, just send a character, e.g. "x" without <CR> and the macro stops executing.

The macro can be started again by "EM n " where n is the number of the macro.

7.1.4 Limitations

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

31 macro sequences (numbers 1 to 31) are available for general use. Macro #0 is special: it is called the *autostart macro* and, if defined, 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 without limitation.

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

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

7.1.5 Macro #0 (autostart macro)

Macro #0 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 #0 may also be used to set up custom parameters, either before manual/computer control is begun or before transferring control to other macros for stand-alone operation.

Macro #0 is defined by the "MD0,xxx" command, where xxx represents a comma-separated list of base commands e.g.:

```
MD0,MR50000,WS100,GH,WS100,RP4<CR>
```

Macro #0 can be erased by:

```
RZ<CR>
```

The contents of macro zero can be read by

```
TZ<CR>
```

7.2 Stand-Alone Operation

Mercury controllers offer the extremely useful feature of *autonomous macro execution*, meaning that positioning tasks can be programmed for execution at power-up, even if there is no PC connected. The macro language supports *conditional command execution*, which, in conjunction with the digital I/O lines and a small push button box (C-862.PB3), provides virtually unlimited operational flexibility.

7.2.1 Pushbuttons

The C-862.PB3 Push Button Box connects to the Mercury I/O connector and allows to apply a TTL signal at the input lines and indicate by LEDs the status of the output lines.



Fig. 8:C-862.PB3 Pushbutton box with LEDs for Mercury Controllers

7.2.2 Macros under Pushbutton Control

The following examples illustrate some techniques that are useful in macro programming to make the Mercury a stand-alone controller under operator control.

In the descriptions below, parameters that are said to be “programmed” are set before operation from a host PC. Other variable behavior is determined at run time by an operator using the push button box and with no PC connected.

Example 1

Autostart macro, initialization and “endless” loop functionality.

```
[MC00] EM5,EM13
[MC05] BF,DP200,DI,DD200,SV80000,SA400000,FE2,WS100,DH,SV250000
[MC13] EM14,EM15,EM16,EM17,WA20,RP,RP
```

Macro #0	Autostart-macro that is executed automatically at power up, independent of whether a PC is connected or not. Using it as a switch to call one or two other macros makes the program easier to understand and maintain.
Macro #5	Initialization tailored for M-511.PD stages.. It disengages the motor brake, sets the P-I-D parameters and starts a reference search in the correct direction towards the sensor (FE2). When the reference is found, the position is defined as zero and the velocity is set to 250,000 counts/second.
Macro #13	Long-duration repetition macro, At most every 20 ms, the sequence of macros named in the EM commands (14, 15, 16, 17) is executed (their content is not shown here). The RP,RP makes the sequence repeat more than 4 billion times (in our scale, forever).

Example 2

These macros illustrate conditional command execution. The macro executes from left to right but the XN and XF commands cause the macro to exit immediately if the state of the specified input line (button) does not meet the required condition (high for XN, low for XF):

```
[MC15] XN1,XN2,SV30000,WF1,WF2
```

The above macro sets the speed to 30,000 if button #1 and #2 are pressed together. Macro exits when both are released.

```
[MC13] XN1,WA30,XF2,SV10000,WF1
```

The above macro sets speed to 10,000 if button #1 is pressed, unless button 2 is pressed within 30 ms (because that is probably an attempt to press both at once).

```
[MC14] XN2,WA30,XF2,SV20000,WF2
```

Sets the speed to 20,000 if button #2 is pressed unless button 1 is pressed within 30 ms

```
[MC16] XN3,SV70000,CP8,WA200,XN3,SV130000,CP12,WA200,XN3,SV190000,
        CP14,WA200,XN3,SV250000,CP15,WF3
```

Button 3 sets one of 5 speeds, depending on how long it is held in. LEDs indicate the progression. Every 200 ms the speed is reset and the number of LEDs lit is changed accordingly. Note that the most significant bit for CP corresponds to the bottom LED.

Example 3

The following macros illustrate various types of moves that are useful under operator control:

```
[MC17] XN1,MR-9999999,WF1,AB1
```

Button 1 causes a continuous move in the negative direction. The target is out of range, so the move will continue until interrupted with the AB1 command. That command will only be executed after the button initiating the move is released.

```
[MC19] XN2,WS0},MR500
```

Button 2 causes repeated step moves of 500 counts until the button is released. Differs from a continuous move in that the end position will be a multiple of 500 from the start. Placing the WS before the move allows the rest of the loop to complete before the move has finished.

```
[MC18] XN3,MR5000,WF3
```

Button 3 causes a step move of 5000 counts in the positive direction. The WF3 command ensures that there will be only one step per press of the button.

```
[MC18] XN4,CF1,CF2,MA-50000,EM19,CN1,EM20,EMxxx}
```

```
[MC19] XF1,MR10,WS0,RP10000
```

```
[MC20] XF1,CN2
```

Input 4 initiates a move to -50000 followed by a step move of 10 x 10,000 counts that can be interrupted (with a resolution of 10 counts) by a signal on input 1. When the step move stops, digital output #1 is set high; if the move went to the end, then output #2 will also be set. Note that it is no longer possible to return to the macro that called Macro 18 because of the nesting limit. The solution is to chain to the calling macro explicitly with EM.

A multi-dimensional scan can be arranged by properly interconnecting the IO lines of the Mercury controllers controlling the different axes.

Example 4

This large macro set is designed to allow fully autonomous operation in any of four modes. The desired mode is selected by pressing the corresponding button upon start-up; thereafter the buttons function as defined for the mode chosen, as can be seen from the macro descriptions.

- Mode 1: The motor moves at one of two programmed speeds as long as one of the buttons is pressed. The types of motion associated with the four buttons are "positive-fast," "positive-slow," "negative-fast," and "negative-slow."
- Mode 2: The motor moves to pre-defined positions at a programmed speed and acceleration.
- Mode:3: The motor moves by long or short increments at a programmed speed
- Mode 4: Same as mode 2, but with a different speed setting.

```
// Mercury Macro File
```

```
[MC00] EM+9,EM+10
[MC09] BF,DP200,DI,DD200,SV80000,SA400000,FE2,WS100,DH,SV250000
[MC10] EM11,EM12,EM13,EM14
[MC11] XN1,SV70000,EM15
[MC12] XN2,SV250000,EM20
[MC13] XN3,EM26
[MC14] XN4,SV120000,EM20
[MC15] EM16,EM17,EM18,EM19,WA20,RP,RP
[MC16] XN1,MR9999999,WF1,AB1
[MC17] XN2,MR10000,WF2
[MC18] XN3,MR-10000,WF3
[MC19] XN4,MR-9999999,WF4,AB1
[MC20] EM25,EM21,EM22,EM23,EM24,WA100,RP,RP
[MC21] XN1,MA100000,CP1,WF1
[MC22] XN2,MA50000,CP2,WF2
[MC23] XN3,MA-50000,CP4,WF3
[MC24] XN4,MA-100000,CP8,WF4
[MC25] XN2,XN3,GH,CP6,WF2,WF3
[MC26] EM27,EM28,EM29,EM30,WA30,RP,RP
[MC27] XN1,MR4000,WS50,WA20
[MC28] XN2,MR2000,WS50,WA20
[MC29] XN3,MR-2000,WS50,WA20
[MC30] XN4,MR-4000,WS50,WA20
```

These macros can be stored permanently in the Mercury Controller. The data file itself is an ASCII file and can be stored using the file manager of the *MercuryMove* Software.

Macro #0 Autostart-macro that is executed automatically at power up, independent of whether a PC is connected or not. It calls two other macro commands, macro #9 and macro #10.

Overall initialization

Macro #9 Tailored for M-511.PD stages. It disengage the motor brake, sets the P-I-D parameters and starts a reference search in the correct direction towards the sensor (FE2). When the reference is found, the position is defined as zero and the velocity is set to 250,000 counts/s.

Macro #10 Calls the sequence of macros #11,#12,#13 and #14. This is used to find a button pressed at power up, or just after the reference position is found. If a button is just pressed, another macro is called. If not, Mercury goes to normal operation and waits for command input.

Macros that set up and jump to the sections corresponding to each mode

Macro #11 Only executed if button #1 is pressed. If the button is pressed, the velocity is set to 70,000 and macro #15 is executed.

Macro #12 Only executed if button #2 is pressed. If the button is pressed, the velocity is set to 250,000 and macro #20 is executed.

Macro #13 Only executed if button #3 is pressed. If the button is pressed, macro #26 is executed.

Macro #14 Only executed if button #4 is pressed. If the button is pressed, the velocity is set to 120,000 and macro #20 is executed.

Macros used for Mode 1

Macro #15 Long-duration repetition macro, Every 20 ms the sequence of macros #16, #17, #18 and #19 is executed. The RP,RP makes the sequence repeat more than 4 billion times (in our scale forever).

Macro #16 Only executed if button #1 is pressed. As long as the button is pressed, the motor is moved in positive direction.

Macro #17 Only executed if button #2 is pressed. The motor moves 10,000 steps (positive). When the button is released, the macro is exit.

Macro #18 Only executed if button #3 is pressed. The motor moves 10,000 steps (negative). When the button is released, the macro is exit.

Macro #19 Only executed if button #4 is pressed. As long as the button is pressed, the motor is moved in negative direction.

Macros used for modes 2 and 4

Macro #20 Long-duration repetition macro, Every 100 ms the sequence of macros #25, #21, #22, #23 and #24 is executed. The RP,RP makes the sequence repeat more than 4 billion times (in our scale forever).

Macro #21 If button #1 is pressed, the motor moves to position 100,000. The digital output pattern is set to 1 (LED #1 ON). The macro exits when the button is released.

Macro #22 If button #2 is pressed, the motor moves to position 50,000. The digital output pattern is set to 2 (LED #2 ON). The macro exits when the button is released.

Macro #23 If button #3 is pressed, the motor moves to position -50,000. The digital output pattern is set to 4 (LED #3 ON). The macro exits when the button is released.

Macro #24 If button #4 is pressed, the motor moves to position -100,000. The digital output pattern is set to 8 (LED #4 ON). The macro exits when the button is released.

Macro #25 If button #2 and button #3 are pressed at the same time, the motor goes home. The digital output pattern is set to 6 (LED #2 and #3 ON). The macro exits when both buttons are released.

Macros used for Mode 3

Macro #26 Long-duration repetition macro, Every 30 ms the sequence of macros #27, #28, #29 and #30 is executed. The RP,RP makes the sequence repeat more than 4 billion times (in our scale forever).

- Macro #27 If button #1 is pressed, the motor moves 4,000 steps, waits until the position is reached, then waits for another 20 ms and repeats the move as long as the button remains depressed.
- Macro #28 If button #2 is pressed, the motor moves 2,000 steps, waits until the position is reached, then waits for another 20 ms and repeats the move as long as the button remains depressed.
- Macro #29 If button #3 is pressed, the motor moves -2,000 steps, waits until the position is reached, then waits for another 20 ms and repeats the move as long as the button remains depressed.
- Macro #30 If button #4 is pressed, the motor moves -4000 steps, waits until the position is reached, then waits for another 20 ms and repeats the move as long as the button remains depressed.

8 Software

Mercury host software is available on the included PI Motion CD.

Fig. 9:



8.1 Host Software

Mercury controllers can be used with the *MMCRun Host* software, a versatile operating environment for Windows™ 95/98/2000 and NT. This software offers many features for commanding and macro programming the Mercury. Depending on the Mercury firmware version, use this PC Software:

Mercury Firmware Version	PC Software
8.10 and earlier	NetMove Version 4.32
8.40 and later	MMCRun Version 8.01

8.1.1 Start Screen



Fig. 10: MMCRun.EXE Mercury host software start screen

The Mercury host software supports up to 16 controllers in a daisy chain with RS-232 cables. If only one Mercury is connected, select "Single Axis" and choose the device number (*Note: The device number runs from 1 to 16 while the physical address the controller from 0 to 15*).

If multiple Mercury controllers are networked, select either "Network, scan 1 to 4" (if all device numbers are smaller or equal to 4) or "Network, scan 1 to 16" (if any Mercury is set to a device number larger than 4).

If you know that the Mercury controller 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 code or parameter changes will be sent to the controller.

If you are using the Mercury controller the first time and you have mechanics of a special type, choose the corresponding "initialize to" option.

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

8.1.2 Main Screen

The main screen has a large pane which is a sort of desktop on which other program Windows™ can be placed, as well as buttons to open these Windows™ and control communications with the controller(s).

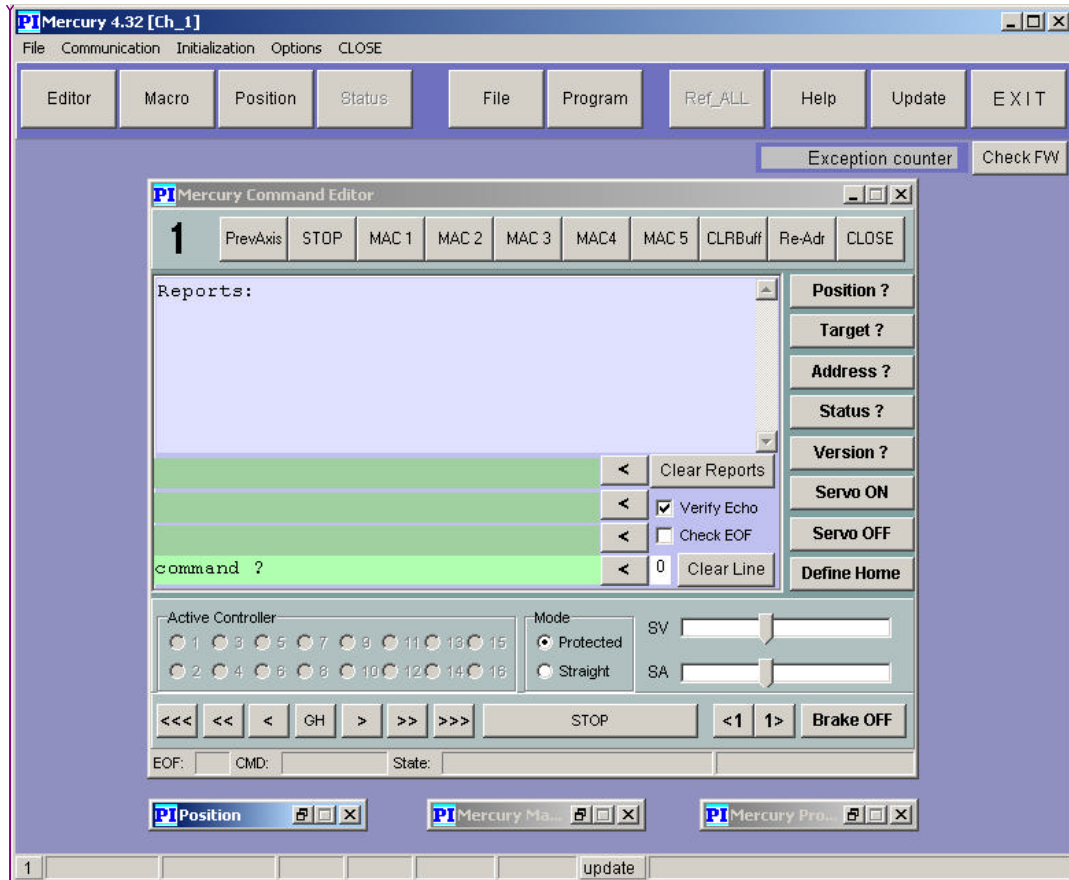



Fig. 11. Main screen as it appears upon entry

The main screen acts as a sort of desktop for the other windows in the program, which can be moved around and resized at will. The main screen opens with the *Command Editor* window open and the *Position*, *Macro Manager* and *Program Editor* windows minimized within it. Click on the toolbar button for any unavailable

window you wish to use (*Editor*, *Macro*, *Position*...). Using the  button next to the title of a minimized window is not recommended.

If a vertical scroll bar appears at the right of the main window, using it is one way to bring a desired window into view.

Controls and Indicators

Editor	Opens the Command Editor window
Macro	Opens the Macro Manager
Position	Opens the real-time position display
Status	
File	
Program	
Ref_ALL	

Help	Shows a list of commands
Update	updates the status line entries
Exit	
Exception counter	Counts bad communication events
Check FW	Checks firmware version

Command Editor Window¹

The *Command Editor* appears open on the main screen (see Fig. 11).

<u>Top Row</u>	
PrevAxis	Switches to previously-active axis
STOP	Immediate motor stop, Motor halt
MACn	Execute macro <i>n</i>
CRLBuff	clears the communication buffer
Re-addr	Send an address-selection command with the address of the controller selected in the <i>Active Controller</i> pane
Close	Close <i>Command Editor</i> Window

<u>Right Side</u>	
Position?	TP Displays current position active axis
Target?	TT Displays programmed target position
Address?	Gives device number (=address+1) of selected controller
Status	TS Displays controller status
Version	VE Displays version information
Servo ON	MN Motor servo-loop ON
Servo OFF	MF Motor servo OFF
Define Home	DH Sets position counter to 0
Brake OFF	BF Sets brake off

<u>Bottom Row</u>	
<<<	MR move relative (-) 200,000 counts
<<	MR move relative (-) 50,000 counts
<	MR move relative (-) 10,000 counts
>>>	MR move relative (+) 200,000 counts
>>	MR move relative (+) 50,000 counts
>	MR move relative (+) 10,000 counts
GH	GH go to zero position
1	MR move one step in negative direction
>1	MR move one step in positive direction

<u>Command Pane</u>	
Clear Reports	clears the report window
Clear Line	clears the command line

<	send command
---	--------------

¹ All functions apply to currently selected (active) controller, unless otherwise noted.

8.1.3 Position Window

The Position window shows the results of the software position polling. Polling of position and position error can be switched on or off by checking or unchecking the corresponding boxes. If the SChar checkbox is checked, then single-character position readout commands are used, with the advantage that polling can continue during macro execution.

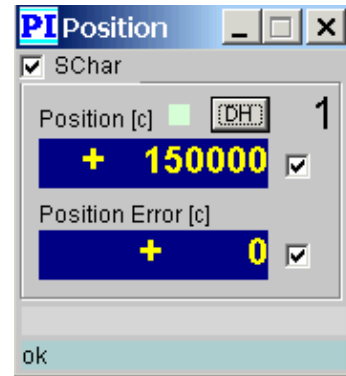
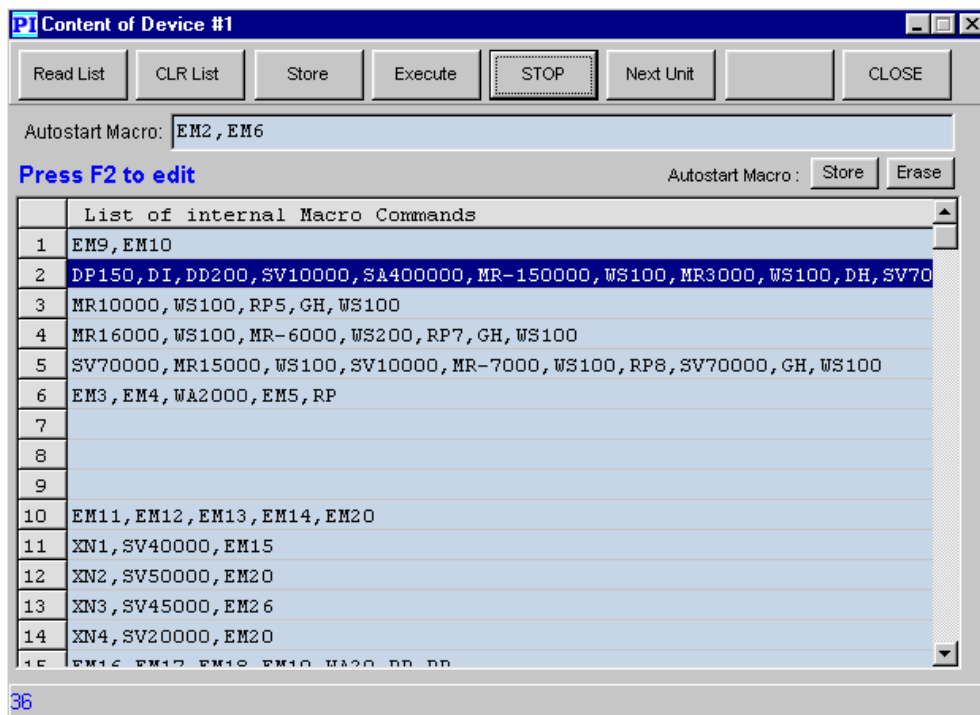


Fig. 12. Position window

8.1.4 Macro Manager

The Macro Manager allows display editing of macro command sequences stored in the Mercury controller.

Choose the macro to edit by clicking the corresponding row, then press the F2 key to activate the edit mode. After changes are made, just press <CR> to send the command line to the controller.



8.1.5 Mercury File Manager

The Mercury File Manager can be used to download, upload and copy macro sequences between the controller and the host PC. On the host, the sequences are stored in files in ASCII text format, so any text editor like *Notepad* can be used to edit the macro files.

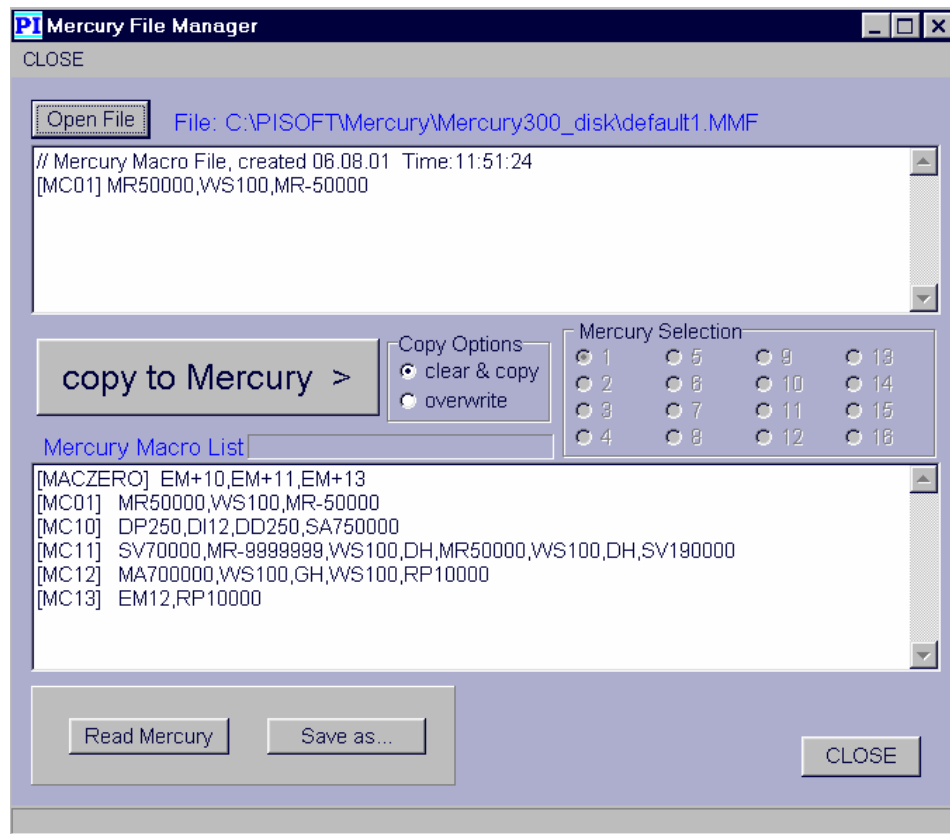


Fig. 13:Macro file manager window

Note that one file can contain more than one macro, the macro number being included in the bracketed expression at the beginning of the line.

The functions of the buttons and indicators are as follows:

The *clear & copy* and *overwrite* buttons affect the way the macros in the working list on the screen are copied to the controller when *copy to Mercury* is pressed. If *clear & copy* is selected, all 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.

8.1.6 Program Editor

The program editor allows to create and manage sequences of commands that are stored in the host computer (rather than in the controller itself).

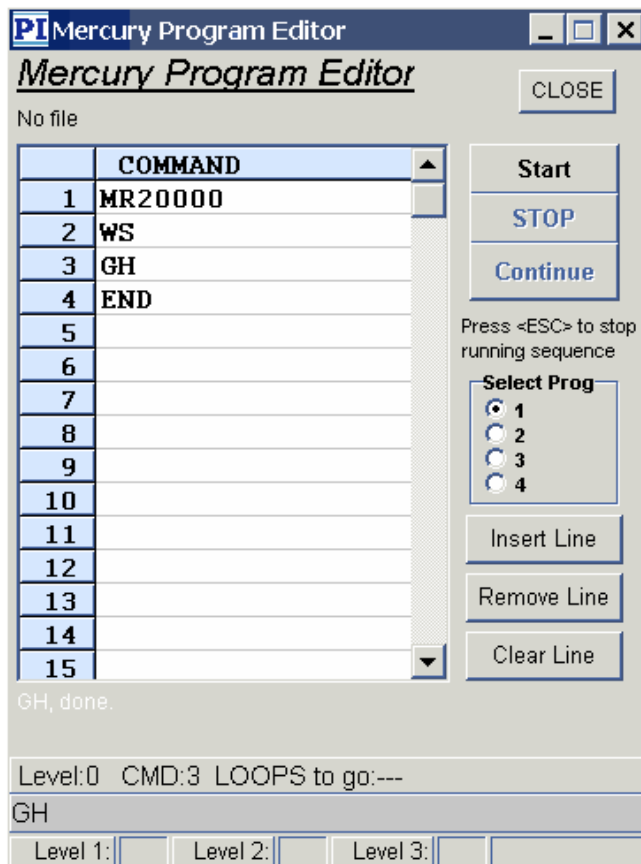


Fig. 14:Program Editor window

The program editor also makes it possible to sequence commands to multiple Mercury controllers. The motion of one controller can be coordinated with the motion status of another. Special commands are available for use with the program editor:

- SC*n*: select controller number *n* (range of *n*: 1..16)
- WS: Wait for stop. This halts command execution until the move of the selected controller has terminated.
- DO: Start a Do-Loop structure
- LOOP*n* End a Do-Loop structure and branch to the corresponding DO until the loop has been executed *n* times. Do-Loop structures can be nested 4 levels deep.

The controls in the window are self-explanatory.

9 Mercury Joystick Control

A pair of networked Mercury controllers can be controlled using a joystick connected to the game port on the host computer. Joystick control is implemented with a separate host software program, *MJoy*.

See the Mercury Joystick Software Manual (MS 88E) for details.

10 Troubleshooting

- Problem (1): The controller communicates and reports position values (TP) , but the connected stage or motor does not move.
- Possible cause:
- (a) Make sure 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 integrated PWM amplifier (.PD stages), verify that the 24 VDC supply is connected to the stage and is operating properly.
- Problem (2): Both LEDs are blinking and the controller does not communicate
- Possible cause: The firmware memory is signaling an internal error. If this problem persists after power-cycling, do a firmware update (or reload the same firmware version).
- Problem (3): Using the Mercury Controller software, the controller seems not to respond to queries, even though other commands are executed.
- Possible cause: Set the echo mode OFF (EF)
- Problem (4): The controller seems to do nothing when it is switched on
- Possible cause:
- (a) The baud rate of the controller and PC may not match. Try different baud rates on the host PC.
 - (b) the controller is not being correctly addressed. Make sure that the address set in DIP switches agrees with that being used by the software that is trying to send commands.
 - (c) There may be an autostart macro stored in the controller. . Sending any command will interrupt the macro, unless that macro falsely addresses the controller (SC command). Once interrupted, the macros can be examined or deleted.
- Problem (5): The controller seems to operate erratically when switched on
- Possible cause: (a) There must be an autostart macro stored in the controller. Proceed as above.
- Problem (6): The controller does not seem to react properly to limit signals from the stage.
- Possible causes:
- (a) Limit detection not set to proper logical level (command LH or LL) or not activated (command LN)
 - (b) Non-PI stage limit signal not compatible with Mercury's limit signal input (e.g. insufficient current source or sink). Contact PI and stage manufacturer to discover who is at fault

11 Command Reference

This section describes commands supported by firmware 8.40 and higher. The commands executed by a Mercury controller come from two sources: the RS-232 input and the Mercury's own macro storage.

All characters arriving on the RS-232 input are passed on to the RS-232 output. This feature is used when more than one Mercury is controlled off a single RS-232 port of the host computer.

11.1 Command and Response Formats

11.1.1 Command Execution

Upon power-up, the Mercury controller executes its startup macro (if any is present) and, unless selected from inside the macro, remains in the *deselected* state. That means it ignores everything on the RS-232 line except for a short-form address-selection code.

Whenever a short-form address-selection code is sent, the controller with the matching address (if any) goes into the *selected* state and all other controllers into the *deselected* state.

Upon reception of any string except an address-selection or single-character command, the *selected* controller will interrupt any running macro and try to execute the string as a command.

Upon reception of a single-character command, the selected controller will execute the command without interrupting any macro or compound command that may be running.

11.1.2 Command Codes

Mercury commands use a more or less mnemonic code of 1 to 3 characters to identify the type of operation; the code is followed by pertinent data value(s), if necessary. For example, "TP" (Tell Position) by itself is adequate to display the motor position, but "MR" (Move Relative) alone would be useless, because the system would not know how far you wished to move. Some commands which take a data value will assume a particular value if none is specified. All commands are checked for acceptability before execution begins. MR, for example, must be followed by a minimum of 1 and a maximum of 9 digits to accommodate the allowable range of motion.

11.1.3 Errors and Reports

Commands do not themselves send error messages, but rather set an error code in the controller, most of which can be interrogated with the TS (Tell Status) command.

Commands fall into two classes, those which evoke responses (*report* commands) and those which do not. *Report* commands report the requested values in a defined format while having no effect on system operation.

Report commands, which cause the Mercury controller to emit a string of data, be it a position, target, help string or other information, are easy to remember as they usually begin with a "T" for "Tell" or "G" for "GET". Some also have single-character forms, e.g. "?" for TP (Tell Position), which can be issued and answered during macro execution without interrupting the macro.

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

Examples:

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

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

11.2 Command Survey (Firmware Version 8.40)

Note that some commands belong to more than one of the logical groups below.

11.2.1 Address Selection

Since Mercurys ignore all commands until selected, the address selection mechanism is most important. Address selection can be accomplished by sending the two address code characters to the controller.

^A (ASCII character 01) + address number

The address selection code is unique in that even *deselected* Mercurys react to it. It is used to *select* a single controller and *deselect* all others that might be connected in a daisy-chain network. (see also chapter "Address Selection code").

Address selection at power up

If the Mercury should be powered up with enabled communication, there is a special command that can be used:

SCn (Select Controller n)

The SC command must be executed by a controller to be effective. Typically, it is placed in the controller's autostart macro (the controller would not execute it from the RS-232 input unless it were already selected, in which case the command would be unnecessary). Selecting more than one controller in a network at startup is not recommended.

11.2.2 Commands in Alphabetic Order

Command	Function	
AB	Abort: Stop motion abruptly	
AB1	Abort: Stop motion smoothly with programmed deceleration	
BF	Set brake OFF	
BN	Set brake ON	
CA	Pulse output for PZT stages, channel A	
CB	Pulse output for PZT stages, channel B	
CF	Channel OFF	
CN	Channel ON	
CP	Channel pattern	
CS	Report checksum	C:
DD	Define d-term (derivative gain)	
DH	Define home	
DI	Define i-term (integral gain)	
DL	Define integration limit	
DP	Define p-term (proportional gain)	
EF	Set Echo OFF	
EM	Execute Macro	
EN	Set Echo ON	
FE	Find edge (find origin position)	
GD	Get d-term	D:
GH	Go home	
GI	Get i-term	I:
GL	Get integration limit	M:
GP	Get p-term	G:
LF	Limit switch operation OFF	
LH	Limit switches active high	
LL	Limits switches active low	
LN	Limit switch operation ON	
MA	Move absolute	
MD	Macro definition	
MF	Motor off	
MN	Motor on	
MR	Move relative	
RM	Reset (erase) macro	
RP	Repeat from beginning of line	
RT	Reset (like power-on reset)	
SA	Set Acceleration	
SC	Select controller	
SM	Set maximum following error	
ST	Stop motion smoothly and move back	
SV	Set Velocity	
TA	Tell analog input value	A:
TB	Tell board address	B:
TC	Tell channel (digital input)	H:

TD	Tell dynamic target	N:
TE	Tell error (distance from target)	E:
TF	Tell profile following error	F:
TI	Tell iteration number	X:
TL	Tell programmed acceleration	L:
TM	Tell macro contents	none
TP	Tell position	P:
TS	Tell status	S:
TT	Tell target position	T:
TV	Tell actual velocity	V:
TY	Tell programmed velocity	Y:
TZ	Tell Macro Zero	none
UD	Update flash	
VE	Display version number	none
WA	Wait absolute time	
WF	Wait channel OFF	
WN	Wait channel ON	
WS	Wait stop	
XF	Execute if channel OFF	
XN	Execute if channel ON	
'	Single Character Command: TP (Tell Position)	39 (0x27)
#	Single Character Command: TC (Tell Channel)	35 (0x23)
%	Single Character Command: TS (Tell Status)	37 (0x25)
?	Single Character Command: TE (Tell Position Error)	63 (0x3F)
(Single Character Command: TF (Tell Profile Error)	40 (0x28)
/	Single Character Command: (Tell LM629 status)	92 (0x5C)
!	Single Character Command: Halt for all members	33 (0x21)

11.2.3 Motion and Sequencing Commands

AB Abort: Stop motion abruptly
AB1 Abort: Stop motion smoothly with programmed deceleration
ST: Stop motion smoothly and move back
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)

11.2.4 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

11.2.5 Report 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:
TA Tell analog input value	A:
TB Tell board address	B:
TC Tell channel (digital input)	H:
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

11.2.6 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 perform a new start
 SC Select controller
 UD Update flash

11.2.7 Macro Commands

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

11.2.8 I/O commands

CN	Channel ON	
CF	Channel OFF	
CP	Channel pattern	
WN	Wait channel ON	
WF	Wait channel OFF	
XN	Execute if channel ON	
XF	Execute if channel OFF	
TA	Tell analog input value	A:
TC	Tell channel (digital input)	H:

11.2.9 Single-Character Commands (Firmware 8.40)

Single-character commands are used to generate a report by sending only one character. This allows faster requests (saving the second character and the CR), but the main advantage of single-character commands is that they can be used during execution of macros or compound commands without interrupting a running sequence.

When a macro is running, e.g. "MR5000,WS100,GH,WS100,RP" it would be stopped if anything except a short-form address selection code or single-character command were received with the controller selected.

The Single-Character Commands in Firmware 8.40 are:

Char	Command	ASCII decimal	ASCII hex	Report generated	Comment
"?"	TE	63	0x3F	E:xxx	(Tell Position Error)
"("	TF	40	0x28	F:xxx	(Tell Following Error)
"%"	TS	37	0x25	S:xxx	(Tell Status)
"#"	TC	35	0x23	H00:xxx	(Tell Channel)
"' "	TP	39	0x27	P:xxx	(Tell Position)
"!"	AB	33	0x21	Motor Stop	(like AB)
"\"	none	92	0x5C	Z:FF	Tell processor status

Note: Single-character commands are sent without any terminator. As soon as such a character is recognized, the report is generated.

11.3 Command Reference in Alphabetic Order

Note: <CR> means "enter" on the keyboard or CR as ASCII code.

AB Abort motion (abruptly)

Stops the motor abruptly.

The motion profile velocity is set immediately to zero and the previously programmed target position is set to the current position.

AB is used for immediate stopping. When the motor moves at high speed, this command will generate a hard mechanical shock to the system, because no deceleration ramps are used. If you want to smoothly stop the motor, use the AB1 command instead.

Example: AB<CR> : Aborts the motion of the motor immediately

See also: AB1, ST

AB1 Abort motion (smoothly)

Stops the motor smoothly with the programmed acceleration.

The motion profile velocity is continuously decreased down to zero. When the motor has come to a stop, the programmed target position is set to the current position.

AB1 is used for smooth stopping. When the motor moves at high speed, this command allows decelerating without mechanical shocks.

Example: AB1<CR> : Stops the motor smoothly

See also: AB, ST

BN Brake ON

Activates the active-low motor brake output line, (motor connector pin 1, set to 0V) allowing the spring-loaded brake, if present and connected, 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

Deactivates the active-low motor brake output line, (motor connector pin 1, set to 5 V) releasing and cocking the spring-loaded brake, if present and connected.

With M-5x1.DD stages, the high signal on line #1 will enable the motor amplifier.

CN_n Channel ON (range of n: 1...4)

Set output channel *n* to ON (high, +5V). No other channels are affected.

Command: CN_n<CR>

Parameter: n indicates the output channel, can be 1,2,3 or 4.
 Report: none

CFn	Channel OFF	(range n: 1..4)
-------------------------	--------------------	-------------------------------------

Set output channel n to OFF (low, 0V). No other channels are affected.

Command: CF n <CR>

Parameter: n indicates the output channel, can be 1,2,3 or 4.

Report: none

Example: "CF3" sets output channel #3 to zero

CPn	Channel Pattern	(range n: 0..15)
-------------------------	------------------------	--------------------------------------

Set digital output channels 1 to 4 according to the digits in the binary representation of value of n . $n=0$ sets all channels to low (0V), $n=15$ sets all channels to high (+5V). $n=3$ sets channels 1 and 2 high, channels 3 and 4 low, and so on.

Command: CP n <CR>

Parameter: n is interpreted as a decimal number representing the bit pattern for output channels 1 to 4.

Report: none

Example: "CP5" set channels 1 and 3 high, 2 and 4 low
 "CP15" set all channels high
 "CP0" set all channels low

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 (sample): "C:50EE E330"

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.

DH	Define Home
-----------	--------------------

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

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

DI*n* Define Integral gain

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.

DL*n* Define Integral Limit

Limits the amount of contribution by the integral gain function.

DP*n* Define Proportional gain

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 <CR> Sets proportional gain of 80

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 the echo for verification of successful transmission.

EF is the required setting when working with the Mercury Host Software.

EM*n* Execute Macro Command *n* (*range n: 1..31*)

Used to run a previously defined macro command. If there is no macro defined for the number *n*, no action will be taken. If the EM command is executed in a macro, control will be returned to the command after EM unless the called macro itself calls another macro.

Example: EM3 <CR> Execute macro #3

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

EN Echo on

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

FE n	Find Edge	(range n : 0..3)
--------	-----------	--------------------

Move the motor until a transition on the reference line is detected. In conjunction with a physical reference switch, FE can be used to move to known (origin) position. The physical origin point is where the reference input line detects a transition from GND to +5V or vice versa. Most of PIs mechanical stages have a reference sensor that can be used with the Mercury feature. Note that the physical position corresponding to the transition will differ slightly when the reference switch is approached from different sides.

The meaning of the parameter n is:

$n=0$: Search starts in positive direction.

$n=1$: Search starts in negative direction.

$n=2$: Search starts in positive direction if the reference line input level is high. Otherwise the search starts in negative direction.

Use this option when referencing M-500 series stages.

$n=3$: Search starts in positive direction if the reference line input level is low. Otherwise the search starts in negative direction.

Examples:

FE0 <CR> causes motor to move in a positive direction until the reference signal 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 <CR> causes the motor to move in a negative direction until the reference input changes state.

FE2 <CR> causes the M-500 stages to move from all starting positions towards the reference point.

FE3 <CR> can be used with stages with inverted reference signal levels, like the M-400 series.

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 very far positive target. 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.

GH	Go Home
----	---------

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

Example:

GH <CR>: Moves motor to zero position.

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"

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 controller to expect both limit switch inputs to be active-low. When a limit switch 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 controller to expect both limit switch inputs to be active-high. When a limit switch input is greater than 3 volts and limits are enabled, motion in the corresponding direction will be terminated.

MAn	Move Absolute	(-1,073,741,824 < n < 1,073,741,823)
-------------------------	----------------------	--

Starts a move to absolute position n .

MA30000 <CR>: Starts the motor to go to position 30000

MDn	Macro Definition	(range n: 1..31)
-------------------------	-------------------------	--------------------------------------

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 for immediate execution.

Examples:

MD1,MR50000,WS100,GH <CR>

Defines macro #1

MD2,TT,TP <CR>

Creates a macro command to Tell Target, then Tell Position and assigns it to number 2.

EM2 <CR>

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

See also: TM, RM, RZ

MRn	Move Relative
-------------------------	----------------------

Initiates a move of relative distance of n counts from the current target position. n may be either a positive or negative number, and is added algebraically to the current target to obtain the new target. The resulting absolute target position must be between + and -1,073,741,823.

Note that if the motor is already in motion when this command is issued, the current move will be interrupted and a move to the new target initiated immediately.

Examples:

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

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

MR2000,WS100,MR-1200 <CR>:

Motor moves 2000 count positive, then -1200 counts negative.

See also: MA

MF	Motor OFF
-----------	------------------

When this command is issued, the motor no longer holds its position actively (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. The motor position is still monitored in the MF status and may be queried (e.g. 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 has been redefined.

Examples: MF <CR> : 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 <CR>

DH,MN <CR>

The DH command is also useful for determining the encoder resolution. Issue the MF command, manually position the motor/encoder at some known position, 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. ? ? stages have incremental encoders with from 2000 to 4000 counts/ rev.

MN	Motor ON
----	----------

Returns the Mercury controller to the normal system control mode, in which it servo-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. The 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 axis to the target, initiating a move unless the target has been redefined (or the motor was not moved).

Example:

MN <CR>

RM	Remove Macro
----	--------------

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

Example: RM <CR> Removes all stored macros

Note: This command requires about 2 seconds for execution. During that time no communication is possible.

RP n	Repeat	(0 < n < 65,535)
--------	--------	--------------------

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 except for a single-character command or address selection code. This character should not be the first character of a new command, because it will be discarded.

Example: TE,WA500,RP99 <CR>

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

RT	Reset
----	-------

Restarts the internal firmware operation, as if from a power-off condition. All values are restored to their defaults. If the autostart macro (Macro 0) exists, it will be executed.

RZ	Remove Macro Zero
----	-------------------

Used to remove the autostart macro (macro 0) from the memory.

Example: RZ <CR> Removes the autostart macro

SAn	Set Acceleration	(200 < n < 1,073,741,823)
-----	------------------	---------------------------

Sets the acceleration rate in encoder counts per second squared.

Typical acceleration values are in the range from 100,000 to 2,000,000. The default value is set to 150,000.

Note:

The acceleration value must no be smaller than 200.

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.

ST	Stop Motion
----	-------------

This command stops the motor smoothly.

At the point in time when this command is received, the controller reads the current motor position and positions the motor smoothly to this position. Depending on the programmed acceleration rate, the motor overshoots and then moves back to the captured position.

The previously programmed target is not changed.

Any command will restart the motor to the previously programmed target. If you want to stay at this position, send a DH or AB

Example: ST <CR> : Stops the motor smoothly and pulls back.

See also: AB, AB1

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

Sets the speed up to which the motor will try to accelerate during subsequent moves. The value *n* 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 drive signal.

Example:

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

TAn Tell Analog Input

Analog values can be read through the input channels 1 to 3 with a resolution of 8 bit.

Parameter: range 0 to 3
n indicates the input channel. if *n*=0 then the state of all 3 channels is reported.

Command: TA1 <CR>
 Report: "A1:0255" CRLFETX

Command: TA0 <CR>
 Report: "A1:0175" CRLF
 "A2:0220" CRLF
 "A3:0044" CRLFETX

TB Tell Board address (Mercury address setting)
--

Reports address of the currently selected Mercury controller. The address is set by DIP switches on the Mercury front panel (see p. 13), address selection is done by command (see p. 18). The default setting is 0

Report: "B:0000"

If the next address selection code sent over the network carries address #1, then the Mercury with address #1 will be selected. Its response to a TB command would be "B:0001".

TCn Tell Channel (range of n: 0 to 4)
--

Reads the digital inputs.

Parameter: channel number, range 1 - 4
 if *n*=0 then the state of all 4 channels is reported as a hexadecimal number from 0 to F with each bit corresponding to the state of one of the channels.(LSB = ch. 1)

Examples:

Command: TC1<CR>
 Report: "H01:1" (input high)
 or
 "H01:0" (input low)

Command: TC0<CR>
 Report: "H00:F" (all inputs high)
 "H00:0" (all inputs low)
 "H00:5" (channels 1+3 high, 2+4 low)

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.

Command: TD <CR>
 Report: "N:+0000126317"

TE Tell Error

Reports the position error of the motor, as determined by subtracting the actual position from the target position. This command will not interrupt motion in progress. 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 <CR>

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. As with any RP loop, the output may be stopped by sending any character except a single-character command or address selection code.

Report: "E:+0000000000"

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 dynamic 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 following error to zero when the obstruction has been overcome. If the condition is not temporary, the following error will typically increase until the programmed limit is reached.

Command: TF <CR>

Report: "F:+0000000117"

TI Tell Iterations

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 0.25 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. Note that this command must be in a repeat loop to be useful: it would otherwise interrupt execution of the loop.

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

Report2 "X:+0000000099" and so on.

TM[n] Tell Macros (0 < n < 127)

Displays all currently stored macro commands. If $n = 0$ or not specified, all macros (except the autostart macro) will be displayed. To display the autostart macro, use the TZ command. Since macros may be defined in any order, the TM command is useful for confirming the existence of, as well as listing the contents of a macro.

Report: "MC001<SP>xxx , xxx , xxx<CR>
<ETX>MC002<SP>xxx , xxx<CR>
<ETX>MC003<SP>xxx , xxx<CR>
<ETX><ETX>"

where <SP>=space, <CR>=carriage return, <ETX>=end of text character

see also: Reset Macro (RM)

TP Tell Position

(Also implemented as single character command)

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 <CR>

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

Report: "P:+0000005555" <CR>
"P:+0000005555" <CR>
"P:+0000005555" <CR> ..(65536 times or until loop interrupted)

TS Tell Status

(Also implemented as single character command)

The TS command reports the status of the system, its motion and limit switch states.

Example: "S:04 A8 04 03 02 00"

First hex digit holds bits 7 through 4, last hex digit holds bits 3 through 0, LSB is bit 0.

1F	LM629 status	Bit 0: Busy Bit 1: Command error Bit 2: Trajectory complete Bit 3: Index pulse received Bit 4: Position limit exceeded Bit 5: Excessive position error Bit 6: Breakpoint reached Bit 7: Motor loop OFF
2F	Internal operation flags	Bit 0: Echo ON Bit 1: Wait in progress Bit 2: Command error Bit 3: Leading zero suppression active Bit 4: Macro command called Bit 5: Leading zero suppression disabled Bit 6: Number mode in effect Bit 7: Board addressed
3F	Motor loop flags	Bit 0: Bit 1: Bit 2: Move direction polarity Bit 3: Move error (MF condition occurred in WS) Bit 4: Bit 5: Bit 6: Move error (Excess following error in WS) Bit 7: Internal LM629 communication in progress
4F	Signal Lines Status	Bit 0: Limit Switch ON Bit 1: Limit switch active state HIGH Bit 2: Find edge operation in progress Bit 3: Brake ON Bit 4: Bit 5: Bit 6: Bit 7:
5F	Signal Lines Inputs	Bit 0: Bit 1: Reference signal input Bit 2: Positive limit signal input Bit 3: Negative limit signal input Bit 4: Bit 5: Bit 6: Bit 7:
6F	Error Codes	(numbers reported): 00: no error 01: command not found 02: First command character was not a letter 05: Character following command was not a digit 06: Value too large 07: Value too small 08: Continuation character was not a comma 09: Command buffer overflow 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.

TT 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. Before reaching the target, it may differ from the dynamic target used by the controller to maximize conformance to the proper motion profile (see the TF command).

If the system is in decimal mode, ten digits will be reported (a leading minus sign, "-" is used for positions on the negative side of the defined "home" position). When the hex mode is in effect, eight hex digits are reported in two's-complement notation.

Command: TT <CR>

Report: "T:+0000000000"

TV n Tell actual Velocity (1 < n < 65,535)

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 <CR>

Report: "V:+0000020006"

TY Tell programmed velocity

Reports the current velocity setting (not the current velocity). This value can be changed with the SV command. During a move, the values reported with the TV and TY commands should differ by only a few counts, except during the acceleration and deceleration phases at the beginning and end of the move.

Command: TY <CR>

Report: "Y:+0000020000"

TZ Tell macro Zero

TZ reads the autostart macro.

The autostart macro, as defined by the "MD0,xxx" command, is automatically executed upon power-on or reset.

Report: "MC000_xxx, xxx, xxx<CR>
<ETX><ETX>"

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

TL Tell Acceleration

Reports acceleration value setting (not the current acceleration). This is the acceleration the controller will try to use at the beginning and end of a move.

Command: TL<CR>

Report: "L:+0000170000"

UD Update Flash Memory

Stores the currently set parameters in 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. The previous values will be lost.

VE Version report

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

transmit: VE <CR>

Report:

```
"(C)2000-42 DIVA Automation, Inc./PI GmbH Karlsruhe & Co. KG,
Ver. 8.40, 13 Jan 2004"
```

WAN Wait Absolute (0 < n < 65,535)

Inserts a wait period of *n* milliseconds before executing the next command.

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

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 <CR>
```

WF n Wait for channel n OFF (range of n : 1...4)

Wait for channel *n* to be OFF.

Can be used for command sequencing. It waits until input channel *n* is OFF before continuing program execution.

Command: WN n <CR>

Parameter: *n* indicates the input channel.

Report: none

W Nn Wait for channel n ON (range of n : 1...4)

Wait for channel *n* to be ON.

Can be used for command sequencing. It waits until input channel *n* is ON before continuing program execution.

Command: W Nn <CR>

Parameter: *n* indicates the input channel.

Report: none

WSn Wait for motor stop

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 <CR>

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 <CR>

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..

XFn Execute if OFF	(range of n : 1...4)
--	------------------------

Execute the remainder of a command line only if input channel n is OFF (0 V).

Command: XF n <CR>

Parameter: n indicates the input channel.

Report: none

XNn Execute if ON	(range of n : 1...4)
---------------------------------------	------------------------

Execute the remainder of a command line only if input channel n is ON (+5V).

Command: XN n <CR>

Parameter: n indicates the input channel.

Report: none

12 Appendix

12.1 Dimensions

Decimal places separated by commas in drawings.

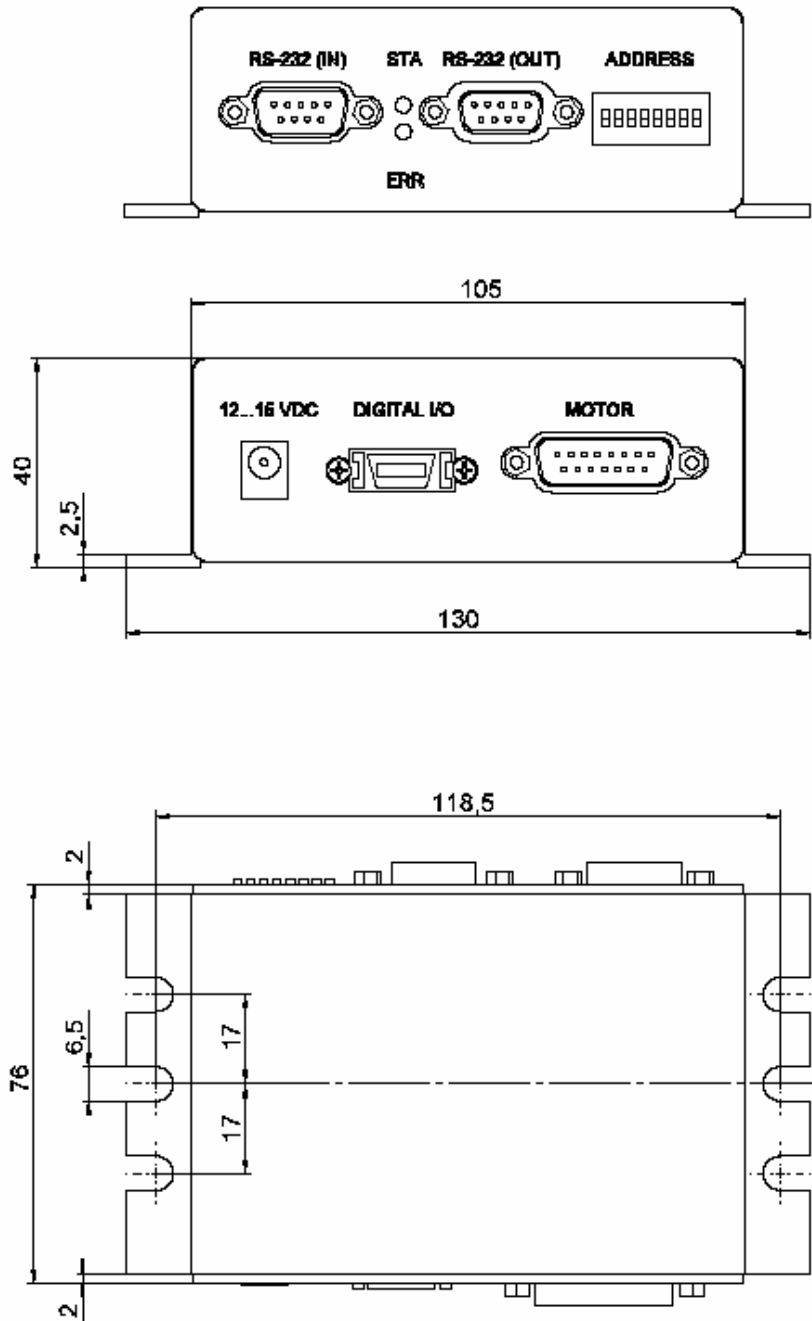


Fig. 15: Merc_dim3.tif

12.2 Motor-Connector Pin Assignment

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

Pin	Function
1	Programmable output (Brake control) (0 or + 5V)
9	(analog) Motor (-) (output)
2	(analog) Motor (+) (output)
10	Power GND
3	PWM magnitude (output)
11	PWM sign (output)
4	+5 V (output)
12	Negative limit signal (input)
5	Positive limit signal (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)

12.3 I/O Connector

Connector "DIGITAL I/O"

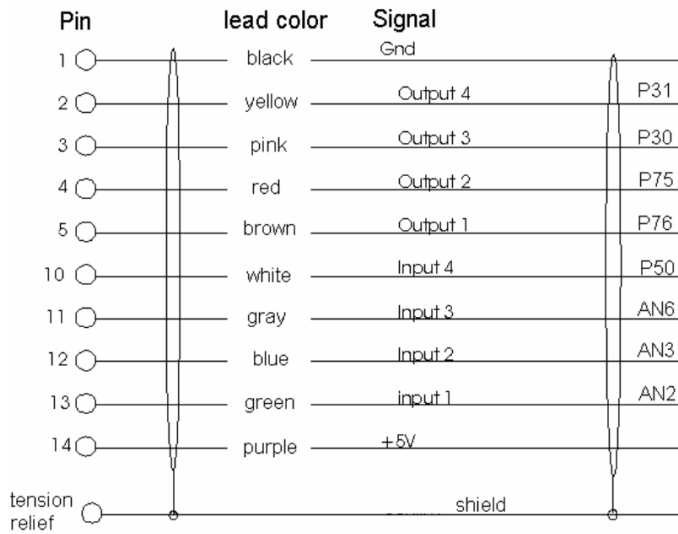
Connector type: Delta Ribbon, 14 pin

Part Numbers

Connector at controller: 3M part number: 10214-52B2JL

Connector at cable: 3M part number: 10114-3000VE

I/O Cable (order# C-862.IO), Pin Assignment



C-862.IO
view at
cable connector

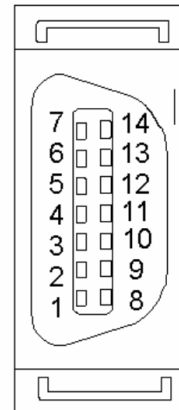


Fig. 16:

