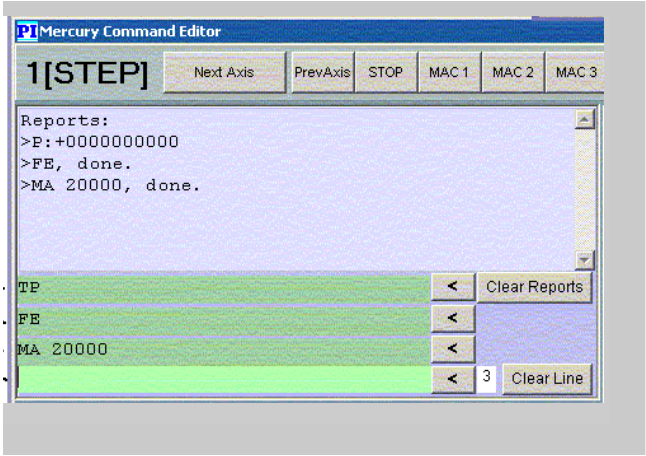
Piezo · Nano · Positioning

MS176E Software Manual

Native Commands

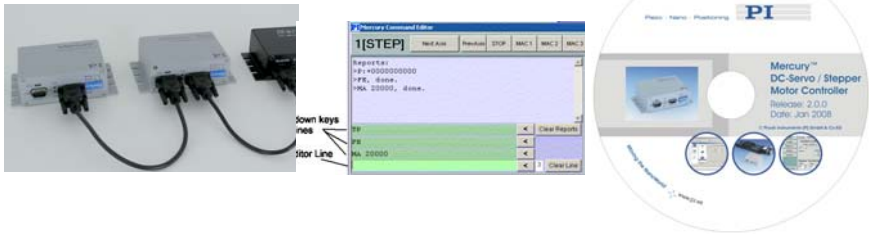
for Mercury™ Class Controllers

Release: 1.0.1 Date: 2007-12-19



This document describes software for use with the following products:

- C-663
  Mercury Step™ Networkable Single-Axis Stepper Motor Controller
- C-862
  Mercury™ Networkable Single-Axis DC-Motor Controller
- C-863
  Mercury™ Networkable Single-Axis DC-Motor Controller



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

Document Number MS176E, Release 1.0.1
MercuryNativeCommands_MS176E101.doc

Subject to change without notice. This manual is superseded by any new release. The newest release is available for download at www.pi.ws.

About This Document

Users of This Manual

This manual assumes that the reader has a fundamental understanding of basic servo systems, as well as motion control concepts and applicable safety procedures.
This manual is designed to help the reader operate Mercury™ Class controllers using the native command set, including native controller macros.

This document is available as PDF file on the Mercury product CD. Updated releases are available for download from www.pi.ws or by email: contact your Physik Instrumente Sales Engineer or write info@pi.ws.

Conventions

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

CAUTION

Calls attention to a procedure, practice, or condition which, if not correctly performed or adhered to, could result in damage to equipment.

NOTE

Provides additional information or application hints.

Related Documents

The Mercury™ controller and the software tools which might be delivered with the controller are described in their own manuals (see below). All documents are available as PDF files via download from the PI Website (http://www.pi.ws/).. For updated releases or other versions contact your Physik Instrumente Sales Engineer or write info@pi.ws.

User Manuals for hardware	Give dimensions, connections and specifications of the hardware components
MMCRun MS139E	Mercury Operating Software (native commands)
Mercury Native DLL & LabVIEW MS177E	Windows DLL Library and LabView VIs (native-command-based)
Mercury GCSLabVIEW_MS149E	LabView VIs based on PI GCS command set
Mercury GCS DLL_MS154E	Windows DLL Library (GCS commands)
PIMikroMove User Manual SM148E	PIMikroMove® Operating Software (GCS-based)
Mercury Commands MS163E	Mercury™ GCS Command descriptions
PIStageEditor _SM144E	Software for managing GCS stage-data database

1 Introduction

Mercury™ Class controllers include the C-663 Mercury Step™ open-loop, stepper motor controller as well as the C-862 and C-863 Mercury™ DC-motor servo-controllers. The C-170 Redstone piezomotor controller is also closely related but is no longer active.

With current firmware, it is possible to operate Mercury™ controllers with two ASCII command sets: the native command set and the PI General Command Set (GCS) . GCS support is currently provided via a Windows DLL which translates GCS-command-based function calls to the native commands. Either command set can be used to set operating modes, transfer motion parameters and to query system and motion values. See the Mercury™ GCS Command manual, MS163E, for a description of the GCS command set.

This manual covers only the native command set.

1.1 Available Native-Command Software

The native ASCII command set is understood by the current Mercury™ firmware directly. With Mercury™ Class controllers, all motion of the connected motors and mechanical stages is software controlled. To offer maximum flexibility, software interfaces at a number of different levels are provided and documented. Most of the individual programs and driver libraries are described in separate manuals. Updated releases are available on the PI Website or via email: contact your PI Sales Engineer or write info@pi.ws.

- *PITerminal* is a Windows program which can be used as a simple terminal with almost all PI controllers. It supports both direct and via-GCS-DLL connection to Mercury controllers via RS-232 and USB (the USB link also looks like a COM port to host software when the USB drivers are installed and the connection is active). *PITerminal* also handles controller selection (by device number) in a Mercury™ network. When used without the GCS DLL, firmware-native commands can be used with the connected controllers .

- *MMCRun* (operating software for Windows™ 95/98/2000/XP, NT and Vista) is the operating software for C-863 and C-862 Mercury™ and C-663 Mercury™ Step controllers. *MMCRun* allows immediate operation of the motion system. It features easy commanding and macro programming of Mercury™ Class controllers. See MS139 for a full description.

- MMC410.DLL: Windows DLL (see MS177 for details) facilitates many interfacing operations and data conversion tasks. Based on native command set.

- LabVIEW™ VIs: facilitates integrating these controllers in the LabVIEW™ environment. Uses the native-command MMC410.DLL above. (See SM177 for details)

1.1.1 Software Installation

If you wish to use any of the GCS-based software, the Windows “Setup” installation procedure on the product CD must be used. (The setup routine will start automatically when the product CD is inserted.) Be sure to log in with administrator privileges to run Setup

If using only the native-command-based software, it is possible to bypass the Setup Wizard.

1 If using the USB interface, the USB driver has to be installed; log on to the PC with administrator privileges. When the Mercury™ is connected and powered up, Windows will discover the new hardware. Follow the on-screen instructions and show the Hardware Wizard the \Driver directory on the Mercury™ CD

NOTE

Windows NT does not have USB support as standard. PI supports only the RS-232 interface for Windows NT. Unless you manage to get USB interfacing operational, the number of networkable devices may be limited to as few as 6 units by the current-sourcing capacity of the PC’s COM port output stages.

2 Copy the contents of the *MMCRun* directory to the host PC. To start MMCRun, double-click the EXE file of the same name

1.2 Control of Multiple Axes

Multiple controllers controlled over the same RS-232 or USB interface are distinguished by hardware addresses (typically set in DIP switches) using an *address selection mechanism*. Note that in other documentation and software, the term “controller address” is sometimes used for the device number (starting with 1) and sometimes for the DIP-switch setting (starting with 0). Commands sent over the interface go the controller which is in the *selected* state and are ignored by the others.

Most native Mercury™ commands begin with a two-letter mnemonic. Because the commands address only the *selected* controller, they do not themselves include controller or axis designators. The syntax of the native commands and a command reference in alphabetical order can be found in Section 4.4.

Up to 16 Mercury™ Class controllers can be networked with each other and controlled over the same RS-232* or USB interface. The controllers interconnect with an RS-232 bus architecture. 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 will continue. Whenever desired,

*The RS-232 output stages of some PCs may not be capable of driving more than 6 units; if this is a problem use USB to interface with the PC.

communication with the selected controller can be concluded and communication with a different controller initiated.

Switching between controllers requires sending an a*ddress selection code*, consisting of two characters (see Section 2.4.2 on p.8). Each controller compares the address in the *address selection code* 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 selected. This means that all controllers connected can execute individual macro commands at the same time. There is no direct communication from controller to controller (unless the digital I/O lines are interconnected and suitably programmed), only from controller to the PC. The host program must handle address selection and motion sequencing among different controllers. This communication model imposes certain 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 code* will continue to be executed, except for commands that issue reports over the link. In this manner, each Mercury™ on the bus can be selected, programmed to execute a desired operation or sequence of operations, then deselected. The same or a different command sequence can then be sent to another controller. See the "Networking" section, p. 9 for programming examples.

1.3 Input/Output Lines

Mercury™ Class controllers offer 4 digital outputs and 3 or 4 inputs for either digital or (8-bit) analog use.

A set of commands is available to handle these I/O lines:

CN, CF, CP, TC, TA, WN, WF, XN, XF (for descriptions see the Native Command Reference Section beginning on p. 17).

Note that there are commands which allow conditional exits from macros depending on the digital and joystick inputs. This makes possible pushbutton control and/or interaxis communication in stand-alone mode, i.e. without a PC connection.

2 Operating Notes

2.1 Position Referencing

Incremental encoder signals can detect minute position changes, but are inherently unable to indicate absolute position. A reference signal detection mechanism is used to determine the absolute physical position of the stage. A position reference switch (sometimes called “origin sensor”) working in conjunction with a capture mechanism of the motion processor can provide this information.

The FE, FE1, FE2 and FE3 (find rising/falling edge) commands are used to start a reference signal search run:

Command	Start the reference position search in:
FE	positive direction
FE1	negative direction
FE2	Automatic reference search for PI stages
FE3	Automatic reference search for older PI stages

2.2 Units in Native Commands

The native command set uses *counts* or *steps* to measure linear or angular distances. Their length in physical units depends on the hardware attached to the controller. Counts are defined by the integrated encoders used for position feedback; steps refer to steps commanded by stepper motor controllers (PI stepper motors are not generally equipped with encoders). Conversion values are included with the technical data specification of the connected stage.

2.3 Testing Communication

Initial communications tests can be performed with either *MMCRun* (if the *Edition* window opens, communication has been established, see document MS139E) or PITerminal (on product CD).

If you are using your own communication routine, first send the proper *address selection code* to “wake up” the desired device*, then send the TP command in order to generate a report string like **"P:+0000000012"**

If you get such a response, the controller system is working properly. Upon startup with no autostart macro, all registers will have been loaded with their default values. Some tasks can be performed using these defaults, but they may need to be modified for some applications.

If you do not receive such a response, read the Troubleshooting Section here and in the hardware User Manual.

2.4 Operating Motors and Stages

Make sure the controller has been “woken up” by an *address selection code* (from the host PC) or an SC command executed from the autostart macro. This is necessary even if there is only one controller on the interface. PI software takes care of this detail for you automatically.

The position counter in the software and firmware is set to 0 upon power-up or reboot. The actual position the axis occupies is not known to the controller. If absolute positioning is necessary, then it is necessary to drive the motor to a known position with a command like FE (see “Position Referencing,” p. 6).

2.4.1 Controller Selection (Addressing)

Mercury™ controllers are in the *deselected* state after power up or reset. In this state they will not execute or respond to any commands. Therefore, the first thing to do is to select a controller. This can be done by sending an *address selection code* over the command interface*, or by placing an SC (Select Controller) command in the autostart macro of one of the units.

Selection from host

Upon receiving an *address selection code* from the host, the addressed unit will enter the *selected* state and all others remain in or enter the *deselected* state. This technique can be used at startup to ensure that one and only one controller is selected, or at any time to stop commanding one unit and begin commanding another (running macros or compound commands on the deselected unit continue to execute, but cease to generate report strings).

Automatic selection at power up

* An address selection code consists of 2 characters: an ASCII decimal 1 followed by an ASCII “numeral” from 0-9 or A-F (decimal 48-57 and 65-70) for the device address of 0-15.

* The address selection code consists of two characters, the second of which specifies the device address of the unit to be commanded. 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 the "Address Selection Code" section).

If a Mercury™ is to be ready to execute commands immediately following power-up or reset, the following command can be executed from the autostart macro of that controller:

SC*n*

Select Controller *n*, where *n* is the address of the unit in question.

Note that executing a command like SC3 from the autostart macros of all controllers on the network will result in only the controller with address 3 being selected. You can thus determine which controller is *selected* at power-up by resetting the address and power-cycling the affected controllers.

Sending commands from the host PC to a network where more than one controller is selected can have unpredictable results. Note that controllers selected with the SC command will also be deselected if the host sends an *address selection code* with a non-matching address.

2.4.2 Address Selection Codes

The *address selection code* 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, denominating the Board Number (Device Number minus 1) from 0 to 15. Note that no termination character should be used.

Board Number (DIP Switch Setting)	Device Number	Address Selection Code	
		1st character	2nd character
0	1	0x01 (decimal 1)	0x30 (decimal 48), “0”
1	2	0x01 (decimal 1)	0x31 (decimal 49), “1”
2	3	0x01 (decimal 1)	0x32 (decimal 50), “2”
3	4	0x01 (decimal 1)	0x33 (decimal 51), “3”
4	5	0x01 (decimal 1)	0x34 (decimal 52), “4”
5	6	0x01 (decimal 1)	0x35 (decimal 53), “5”
6	7	0x01 (decimal 1)	0x36 (decimal 54), “6”
7	8	0x01 (decimal 1)	0x37 (decimal 55), “7”
8	9	0x01 (decimal 1)	0x38 (decimal 56), “8”
9	10	0x01 (decimal 1)	0x39 (decimal 57), “9”
10(A)	11	0x01 (decimal 1)	0x41 (decimal 65), “A”
11(B)	12	0x01 (decimal 1)	0x42 (decimal 66), “B”
12(C)	13	0x01 (decimal 1)	0x43 (decimal 67), “C”
13(D)	14	0x01 (decimal 1)	0x44 (decimal 68), “D”
14(E)	15	0x01 (decimal 1)	0x45 (decimal 69), “E”
15(F)	16	0x01 (decimal 1)	0x46 (decimal 70), “F”

2.5 Networking

Up to 16 Mercury™ Class controllers can be controlled from a single host computer interface.* Communication on the interface is always between the host computer and a *selected* Mercury™, with the other Mercurys™ in the *deselected* state.

Mercury™ Class controllers are in the *deselected* state upon power-up or reset, although the autostart macro (if any) will be executed (and could change the state). In the *deselected* state, they do not execute or respond to any commands on the interface. The host computer can select a particular controller by sending a 2-character *address selection code* over the communications link. This method is often used even if there is only one Mercury™ controller connected. Alternatively, an SC (Select Controller, see Section 4.2 on p. 19) command in the autostart macro (Section .5.2.5) on , p.47) of one of the Mercurys™ can be used to select that unit.

On Mercury™ Class controllers, the Device Number is typically set by DIP switches. Each controller in a network must have a unique address.

After sending the *address selection code*, you can verify 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 (Board Number). If there is no response, then the address did not match that of any Mercury™ controller connected to the network.

2.6 Joystick Control

2.6.1 Handling

CAUTION

Do not enable a joystick axis here when no joystick is connected to the controller hardware. Otherwise the corresponding controller axis may start moving and could damage your application setup.

Most Mercury™ Class motor controllers offer convenient manual motion control by using analog joysticks. With the joystick mode enabled, both *selected* and *deselected* Mercury™s can be joystick-operated. The two joystick axes can be connected through the C-819.20Y cable to two compatible Mercury™ controllers.

When a joystick is connected directly to a controller (not to the host PC), it is the *velocity* (not the position or motion of the target) of the controlled axes that is determined by the joystick position.

* The RS-232 output stages of some PCs may not be capable of driving more than 6 units; if this is a problem use USB to interface with the PC.

NOTE

If the Y-cable is used to connect a joystick to two controllers, the X-branch must be connected to a powered-up controller because joystick power is taken from that side.

NOTE

Before a joystick can be operated correctly, a calibration routine may need to be performed. Activating the joystick before calibration may cause the motor to start moving even though the joystick is in the neutral position. To calibrate a joystick axis turn the corresponding “Adjust“ knob on the joystick until the motor stops. If using *MMCRun* this procedure is facilitated by clicking *Adjust* in the *Joystick* pane. See the *MMCRun* software manual for details.

Commands for joystick handling:

JNn	Enables the joystick operation with a specified maximum velocity n. Range for n : 1000 to 500000 (unit is counts or steps per second) Example: **JN30000** After this command is sent, the joystick is active. At full tilt angle the motor speed is 30,000 steps/s, as indicated. While joystick is active, move commands are not accepted.
JF	Disables the joystick and reactivates processing of move commands
JTn	Chooses Predefined Joystick Response Table. n can be either 0 or 1 and determines whether the response curve is linear (n=0) or cubic (n=1). The cubic curve offers more sensitive control around the middle position and less sensitivity close to the maximum velocity. As default, the linear table is loaded.
TA5	Read joystick axis position, analog, from 0 to 255
TA6	Read joystick button state, analog, from 0 to 255, high values indicate button pressed
YN YF	Conditionally interrupt compound command based on button state

2.6.2 Joystick Response Definition Tables

Normally the joystick response is defined by loading a predefined response table using the JT command. The loaded response table will remain chosen even when the controller is reset.

For special purposes, it is possible to generate a user-defined joystick response table.

Commands for user-defined joystick tables (see Command Reference section for more details):

SIn	Sets Index pointer for joystick table. A new table value can then be written with the SJ command.
SI?	Report command asking for the current index value
SJn	Sets the table value at active index to n. Range of n : 0 to 1024 Value 1024 represents maximum velocity, 0=standstill.
SJ?	Report command asking for the current table entry.

Typical Command Sequence:

```
SI0
SJ0
SI1
SJ5
SI2
SJ10
SI3
SJ15 etc.
```

2.7 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 pre-wired for operation with compatible Mercury™ controllers.

The Mercury™ controller can be configured to accept either an active-high or an active-low stop signal from the limit sensors (both must be the same).

On the C-663 and C-863, the limit switch signals are interpreted by both the controller hardware and software. DIP switch 7 changes the hardware interlock between active-high (OFF) and active-low (ON).

For the software configuration the following commands are used with all Mercurys™:

LH	Limit switch logic active high	p. 29
LL	Limit switch logic active low	p. 29

In addition, it is possible to disable software limit switch evaluation completely:

LN	Limit switch operation ON	p. 30
LF	Limit switch operation OFF	p. 29

NOTE

If the hardware and software limit switch configurations are not compatible, no motion is possible. Standard PI stepper motor stages have active-low limit switches, whereas PI DC motor stages have active-high limit switches.

3 Native Commands

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

3.1 Command Types

- Address Selection Code: 2 ASCII characters beginning with 01 (ctrl-A); instructs the specified controller to enter the *selected* state and all non-specified controllers to enter the *deselected* state. This is the only command that can be sent over the interface and affect controllers in the (power-up default) *deselected* state.

All the following command types are ignored by controllers in the *deselected* state.

- Base Commands: (e.g. "MR5000")
  Single commands, consisting of a mnemonic optionally followed by an argument. Execution begins immediately after reception of the CR character marking the end of the command.

- Compound Commands (e.g. "MR5000,WS100,GH")
  Series of base commands separated by commas. Sequential execution starts upon receiving the CR, which marks the end of the compound command.

- Single-Character Commands (e.g. 0x27)
  One character without terminator.
  Immediate report or action

- Conditional Commands (e.g. "XN1,MR5000")
  For use in compound commands only; affect the interpretation of the base commands following them in the compound command.

3.1.1 Address Selection Codes

An *address selection code* 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, for the Board Numbers (Device Number minus 1) from 0 to 15. Note that no termination character should be used.

This sequence instructs the specified controller to enter the selected state and all non-specified controllers to enter the *deselected* state. This may occur during execution of another command or macro. It is the only command which can affect controllers in the *deselected* state.

In the *deselected* state, all commands except address selection codes are ignored, and all reports (e.g. from running macros) surpressed.

Because Mercurys™ power-up in the *deselected* state, an address selection code is typically the first sequence sent. See also the SC command, for selecting a controller from within its autostart macro.

3.1.2 Base Commands

A *base command* is one command followed by a terminating CR (ASCII character 10). Note that PI GCS software uses LF as command terminator. PI software, including PITerminal, automatically switches termination character depending on the controller attached.

The command is executed immediately after the CR is received. The command will be repeated once for each CR that is received immediately after it is entered. This feature makes it very easy to continuously execute the same command by simply holding down the Enter key. Both uppercase and lowercase characters are valid, and spaces are allowed.

Examples of base commands:

MR2000	Move motor 2,000 counts/steps relative to the present target
DP250	Set p-term of servo filter to 250
MN	Set motor in ON state
GH	Send motor to home position (go home)
MA20000	Send motor to absolute position 20,000
TP	Reports position as: "P:+0000000000"

3.1.3 Compound Commands

A *compound command* is a series of base commands separated by commas. It is thus possible to string together several commands before terminating them with a CR (Carriage Return).These base commands will then be executed sequentially.

The syntax for a compound command is:

CMD[*n*], CMD[*n*], ...,..., CR

Examples:

```
MR2000,WS100,MR-4300
mr5000,ws100,wa500,ma12000,ws100,wa800,tp
MA3500, WS100,WA120,tp
```

The compound command in the following example instructs the motor to move 1000 steps/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
```

Once this compound command is entered, it remains in the buffer until replaced by another non-blank line of input and can be re-executed by sending a carriage return CR alone (blank line).

3.1.4 Single-Character Commands

Single-character commands are used to generate a report or carry out an action by sending only one character without a following CR. This allows faster requests (saving the second character and the CR), but the main advantage of single-character commands is that they are executed immediately even when the controller is in the process of executing a macro or compound command, without interrupting a running sequence (except for the “motor stop” command “!”).

Find a complete overview of single-character commands in the Command Reference section on p. 21.

3.1.5 Conditional Commands

Conditional commands are normally used in compound or macro commands. The state of an input line decides weather the following command(s) are executed or not.

Example: **`XN4,MR5000`**

The relative move is only executed if digital input #4 is ON (high).

3.2 Commands with Responses

Some commands cause the controller to send a string of data to the host computer, be it a position, servo-control parameters, help or other information. These commands, also called *report commands,* usually begin with a T (tell) or G (get).

A compound command or macro containing a report command will also issue the corresponding report when the report command is executed. The reports are suppressed, however, if controller is put in the *deselected* state.

3.3 Command Execution Mode

Base or compound commands can be executed in one of two modes:

- Direct mode, where the command comes in to the *selected* controller over the active interface,
- Macro execution mode, where commands in macros stored on the controller are being executed.

The execution mode does not affect the way the controller responds to single-character commands or *address selection codes* sent over the interface. Nor does the selection state affect operations in macro execution mode, except that reports are suppressed when the controller is *deselected*.

3.3.1 Direct Mode

If there is no Macro #0 defined, the controller goes into direct mode upon power up or reset .

There are only two ways to place it in macro execution mode:

- Call a macro (executing an EM*n* command where *n* is a defined macro)
- Make sure Macro #0 is defined and restart (e.g. with RT).

3.3.2 Macro Execution Mode

When in macro execution mode, the controller returns to direct mode under the following circumstances:

- Controller is in *selected* state and receives any character other than single-character command or the two characters of an address selection code
- Controller is power-cycled or reset with no Macro #0 defined
- Macro which called another macro terminates
- Macro not called by another macro terminates

To achieve a full understanding of how this works, see the EM command description.

4 Native Command Reference

This section describes commands supported by firmware 1.06 and higher.

All single commands fall into one of two classes, those which evoke responses (*report* commands) and those which do not. *Report* commands report the requested values in a defined format.

4.1 Command and Response Formats

4.1.1 Command Execution

Upon power-up, all Mercury™ Class controllers execute the startup macro (if present) and, unless selected from inside the macro, remain in the *deselected* state. That means the controller ignores everything on the command interface except for *address selection codes*.

Whenever an *address selection code* is sent, the controller with the matching address goes into the *selected* state and all other controllers into the *deselected* state. The controller stays in the *selected* state as long as no other, non-matching *address selection code* is received.

Upon reception of any character except an *address selection code* or a single-character command, the selected controller will interrupt any running macro or compound command and start to interpret the input as a command.

Upon reception of a single-character command, the selected controller will execute the command immediately, even during execution of a macro or compound command.

4.1.2 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.

NOTE

All commands are terminated by the termination character CR (carriage return, ASCII character decimal 13). Exceptions are single-character commands and address selection codes which are executed immediately without termination.

Examples:

MR5000CR	Move Relative 5000 counts or (micro)steps
TPCR	Tell Position, causes the report to be sent

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.

MR, for example, must be followed by a minimum of 1 and a maximum of 9 digits to accommodate the allowable range of motion.

4.1.3 Report Commands

Report commands, which cause the Mercury™ controller to emit a string of data, be it a position, target or other information usually begin with a “T” for “Tell” or "G" for “GET”.

Single-character commands generate the same report strings as the equivalent base command would:

Example:

"%" equals **TS**CR

Report strings generated by a report command are terminated by the three-character sequence CR LF ETX (ASCII 13, 10 , 03)

Examples of report strings :

Command: TT	Report:	**T:+0000035500** CR LF ETX
Command: TL	Report:	**L:+0000000120** CR LF ETX

NOTE

The command and report terminators (CR LF ETX) will be omitted in most of the examples in this manual

4.2 Base Commands in Alphabetic Order

Com-mand	Function	Report Identifier	Page
AB	Abort: stop motion abruptly		
BF	Set brake OFF (and/or deactivate some motors)		
BN	Set brake ON		
CF	Channel OFF		
CNn	Channel ON		
CP	Channel pattern defining digital outputs		
CS	Report checksum	C:	
DCn	Define drive current		24
DH	Define home		
EM	Execute macro		
FE	Find edge (find reference position)		
GH	Go home		
HC	Set hold current		
HT	Set hold time		
JF	Set joystick OFF		
JN	Set joystick ON		
JT	Select joystick table		
LF	Limit switch operation OFF		
LH	Limit switch logic active high		
LL	Limit switch logic active low		
LN	Limit switch operation ON		
MA	Move absolute		
MD	Macro definition		
MF	Motor servo off		
MN	Motor servo on		
MR	Move relative		
RM	Remove (erase) specified macro		
RMALL	Remove all macros and parameters		
RP	Repeat from beginning of line		
RT	Reset system (like power-on reset)		
RZ	Remove (erase) Macro 0 (autostart macro)		
SA	Set acceleration		
SC	Select controller		
SI	Select Index for Joystick table		
SJ	Set joystick table value		

Com-mand	Function	Report Identifier	Page
ST	Stop motion smoothly and move back		
SV	Set velocity		
TA	Tell analog input value	An:	
TB	Tell board address	B:	
TC	Tell channel (digital input)	H0n:	
TI	Tell iteration number	X:	
TL	Tell programmed acceleration	L:	
TM	Tell macro contents		
TP	Tell position	P:	
TS	Tell status	S:	
TT	Tell target position	T:	
TV	Tell current velocity	V:	
TY	Tell programmed velocity	Y:	
TZ	Tell macro zero		
VE	Display version number		
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		
YF	Execute if connected joystick button is OFF		
YN	Execute if connected joystick button is ON		

4.3 Single-Character Commands

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

Char	Equivalent Command	ASCII decimal	ASCII hex	Report generated	Comment
"+"	**TE**	43	0x2B	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	(none)	Motor stop
"\"	**(none)**	92	0x5C	0 or 1	moving status
"/"	**TA2**	47	0x2F	A2:xxxx	Tell Analog 2
"&"	**TA1**	38	0x26	A1:xxxx	Tell Analog 1
")"	**TA4**	41	0x29	A4:xxxx	Tell Analog 4

*some commands not available for C-862

**some commands not available for C-663 Mercury™ Step

NOTE

Single-character commands are sent without any terminator. As soon as such a character is recognized, the command is executed.

4.4 Base Command Reference in Alphabetic Order

NOTE

All commands have to be sent with the termination character CR (carriage return)

All report strings are terminated by the sequence: CR LF ETX (Carriage Return + Line Feed + EndOfText)

AB Abort motion

Stops the motor immediately.

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. If running at high speeds you may take into account that stopping the motor abruptly may cause the motor to lose track of the position. Use the ST command if you want to stop the motor smoothly.

Example:

AB	Aborts the motion of the motor immediately

See also: "!" (single-character command) and ST

BF Brake OFF

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

With many .DD and .PD stages, the high signal on line #1 (brake off) is required to enable the motor amplifier, whether the stage is equipped with a brake or not.

BN Brake ON

Activates the active-low motor brake output line, (motor connector pin 13, set to 0V) causing a motor brake, if present and connected, to engage. The factory default power-up state is 0 V (brake on).

CF*n* Channel OFF (range n: 1..4)

Sets digital I/O output channel n to OFF (low, 0V). No other channels are affected.

Command: CF*n*

Parameter: *n* indicates the digital I/O output channel, can be 1,2,3 or 4.

Report: none

see also CN, CP

Example:

CF3	sets digital I/O output channel #3 to zero

CN*n* Channel ON (range n: 1...4)

Sets digital I/O output channel *n* to ON (high, +5V). No other channels are affected.

Command: CN*n*

Parameter: *n* indicates the digital I/O output channel, can be 1,2,3 or 4.

Report: none

See also: CF, CP

CP*n* Channel Pattern (range n: 0 to 15)

Sets digital output channels 1 to 4 according to the digits in the binary representation of value of *n* (bitmapped). *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*

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

Report: none

see also: CN,CF

Examples:

CP5	Sets channels 1 and 3 high, 2 and 4 low (5 = 0101_b)
CP15	Sets all channels high
CP0	Sets all channels low

CS CheckSum

Reports the current firmware version.

Command	Report
CS	**C:0106**

DCn Define Drive Current

C-663 Mercury Step™ only: sets the motor phase current (drive current) for moving state.

Parameter n : Current in mA

Example:

DC400	Sets the drive current to 400 mA.

See also: HC, HT

DD*n* Define Derivative gain (0 < n < 32,767)

DC-motor versions only: 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*n* Define Home

Defines the current motor position as *n* (home position will be *n* counts/steps away).

Example:

DH	Defines the current position as 0.
DH20000	Defines the current position as 20000

DIn Define Integral gain

DC-motor versions only: 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.

DLn Define Integral Limit

DC-motor versions only: Limits the amount of contribution by the integral gain function.

DPn Define Proportional gain

DC-motor versions only: 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:

DP22080CR	Sets proportional gain of 22080

EM*n* Execute Macro Command *n* (range *n*: 1 to 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	Execute macro #3

See also: MD

FE*n* Find Edge (range *n*: 0 tp 3)

Searches for the reference (origin) position.

The motor moves until a transition on the reference line is detected. In conjunction with a physical reference switch, FE can be used to move to a 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 this 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:	Auto-Referencing Option: With standard PI stages search starts in correct direction towards reference point
n=3:	Auto-Referencing Option for stages with non-standard reference signal level

Examples:

FE0	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	Causes the motor to move in a negative direction until the reference input changes state.

GH Go Home

Causes the motor to move to the currently defined zero position. Equivalent to an MA0 (Move to zero position) command.

Example:

GH	Moves motor to zero position.

HC Set Hold Current

Stepper motor versions only: Sets the hold current for the stepper motor.

The hold current becomes active after a programmable time after a move has terminated. Normally the hold current is about 25% of the drive current and this allows to keep the temperature of the stepper motor down, close to room temperature.

Example:

HC250	Set hold current to 250 mA

See also: DC, HT

HT Set Hold Time

Stepper motor versions only: Sets the hold time, that is the delay time between completion of a move and the activation of the hold current.

Example:

HT500	Set hold time to 500 ms

See also: DC, HC

GP Get Proportional term

DC motor versions only: Reports the current Proportional Gain value. This value can be changed with the DP (Define Proportional Gain) command.

Report: "G:+0000000080"

GL Get Integration Limit

DC motor versions only: Reports the current Integration Limit value. This value can be changed with the DL (Define Integration Limit) command.

Report: "M:+0000002000"

JF Joystick OFF

Disables joystick operation and enables normal command mode.

Example:

JF	**Set Joystick OFF**

See also: JN

JN Joystick ON

Enables Joystick operation.

The parameter n represents the maximum joystick velocity in counts or (micro)steps/s. The joystick axis position is then used to set the motor velocity.

Caution: Do not enable a joystick axis here when no joystick is connected to the controller hardware. Otherwise the corresponding controller axis may start moving and could damage your application setup.

Example (Mercury™ Step):

JN20000	Sets Joystick ON with a maximum velocity of 20,000 (micro)steps/s

See also: JF,JT

JTn Loads Joystick Table

Selects a joystick sensitivity profile.

Example:

JT	Loads Joystick Table with linear values
JT0	Loads Joystick Table with linear (same as JT)
JT1	Loads Joystick Table with cubic values

See also: JN,JF

LF Limit switch OFF

Disables software limit switch operation.

The LN and LF commands affect only the software. The LF command should only be used when hardware limit switches are not installed.

This does not affect the hardware interlock in the controller circuitry, if present (C-663 and C-863). Make sure the hardware limit switch polarity is properly set on the controller: DIP switch 7 ON for active-low, OFF for active-high. PI DC motor drives typically have active-high limit switches, stepper motor drives, active-low. See the hardware User manuals for complete information

See also: LN,LL,LH

LH Limit switch active HIGH

Sets the controller to expect both limit switch inputs to be active-high (PI DC motor drives typically have active-high limit switches). When a limit switch input is greater than 3 V and limits are enabled, motion in the corresponding direction will be terminated.

Make sure the hardware limit switch polarity is properly set on the controller: Typically DIP switch 7 ON for active-low, OFF for active-high.

See also: LL,LN,LF

LL Limit switch active LOW

Sets the controller to expect both limit switch inputs to be active-low (PI stepper motor drives typically have active-low limit switches). When a limit switch input is less than 1 volt and limits are enabled, motion in the corresponding direction will be terminated.

Make sure the hardware limit switch polarity is properly set on the controller: Typically DIP switch 7 ON for active-low, OFF for active-high.

See also: LH,LN,LF

LN Limit switch operation ON

Enables 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.

This does not affect the hardware interlock in the controller circuitry, if present (C-663 and C-863). Make sure the hardware limit switch polarity is properly set on the controller: DIP switch 7 ON for active-low, OFF for active-high. PI DC motor drives typically have active-high limit switches, stepper motor drives, active-low. See the hardware User manuals for complete information

See also: LF,LH,LL

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

Starts a move to absolute position *n*.

Motion is counted in counts or (micro)steps.

Example:

MA30000	Starts the motor to go to position 30000

MD*n* Macro Definition (range n: 1 to 31)

Defines a new macro command. Defining more than one macro with the same value of *n* will result in the loss of all but the last macro so defined. To define a macro, use MD followed by the desired macro number and a comma, and then write the command (base or compound command).

Examples:

MD1,MR50000,WS100,GH	Defines macro #1
MD2,TT,TP	Creates a macro command to Tell Target, then Tell Position and assigns it to number 2.
EM2	Executes macro #2, (Same as entering TT,TP)

See also: TM, RM, RZ

MF Motor OFF

Shuts off the motor current.

On DC motor versions this command turns the servo loop off; the motor no longer holds its position actively and may be moved freely. The MF command is used to prevent unwanted movement (servo dither) 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).

Be careful when using this command on stepper motor versions because the controller may lose track of the motor position.

Examples:

MFCR	Sets motor servo-control to OFF

See also : MN (Motor ON)

Servo control can be enabled again anytime by MN.

MN Motor ON

Sets the motor current back to the hold value. On DC motor versions, automatically sets the target to the current position and reenables servo control, so jerk-free activation of the servo loop is possible.

Note: stepper motor versions may have lost track of the position if they were in the MF state.

Examples:

MN	Sets motor servo-control to ON

See also : MF (Motor OFF)

MR*n* Move Relative

Initiates a move of relative distance of *n* counts/steps 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.
Motion is counted in encoder counts (DC motor versions) or (micro)steps (stepper versions).

Examples:

MR5000	Motor moves 5000 cts or steps in positive direction
MR-330	Motor moves 330 cts or steps in negative direction
MR2000,WS100,MR-1200	Motor moves 2000 cts or steps in positive, then 1200 in negative direction

See also: MA

RM Remove Macro

Used to initialize the memory reserved for one or all macro commands.

Clears (erases) one or all macros.

Example:

RM	Removes all stored macros except Macro 0
RMn	Removes macro n

RMALL Remove Macros and Parameters

Removes all macros stored, including Macro 0, and resets all parameters to their start-up values.

Example:

RMALL	Removes all macros and resets parameters

RP*n* Repeat n times

Causes the command string to repeat *n* times.

Limitation: n <= 32568

If *n* is not specified, the command(s) in the string repeats indefinitely.

The repeat loop may be interrupted by sending any character except for a single-character command or a 2-character address selection code.

Example:

TP,WA250,RP12	The command TP (tell position) is repeated 12 times.

RT Reset

Restarts the internal firmware operation, as if from a power-off condition.

All parameter values are restored to their defaults. If the autostart macro (Macro #0) is defined, it will be executed.

RZ Remove Macro Zero

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

Example:

RZ	Removes the autostart macro

SA*n* Set Acceleration

Sets the acceleration in counts or (micro)steps per second squared.

Typical acceleration values:

C-663: 10,000 to 100,000; default 20,000 steps

C-863, C-862: 50,000 to 2,000,000; default 400,000 counts

Example:

SA60000	Sets the acceleration to 60,000 counts or steps / s^2

SC*n* Select Controller

Puts the controller in the *selected* or *deselected* state

If *n* is the Board Number (Device Number minus 1) of the controller on which the command is executed, the controller is placed in the *selected* state.

If this command is used in a host macro in *MMCRun*, it causes the host PC to place the controller with address *n* in the *selected* state and all other controllers in the *deselected* state.

This command is designed for use in the autostart macro.

Example:

SC0	If run on the controller with address 0, puts it in the selected state

MD0,SC0	Assuming the controller's address is 0, then after power on, then after executing this command, then after future power-ups or resets, the device will ready to communicate without waiting for its address selection code.

SI*n* Set Joystick Table Index

For user-defined joystick tables, the table value for each table index (0 to 255) can be set by command. With the SI command the table index is defined and the next write command (SJ) will write to that index location.

Example:

SI200	Sets the table index to 200
SI?	Report: **I0:200**

SJ*n* Set Joystick Table Value

For user defined joystick tables, the table value can be written to the index position. Table values range from 0 to 1023.

With the **SJ?** command, the table value can be read.

Example:

SJ850	Sets the table value to 850
SJ?	Report: **J0:+850**

SM*n* 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

Stops the motor smoothly.

The controller reads the motor position when the command is received and moves the motor smoothly to this position using the current acceleration values.

Depending on the programmed acceleration rate, the motor will overshoot and then move back to the captured position.

The target is set to the current 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	Stops the motor smoothly and pulls back.

See also: AB

SV*n* Set Velocity

Sets the speed to which the motor will be accelerated during subsequent moves. The value *n* is given in counts or (micor)steps per second.

On a DC motor, if the load changes, the controller attempts to maintain the velocity by varying the motor drive signal.

Example:

SV60000	Sets the velocity of motion to 60,000 counts or steps per second.

TA*n* Tell Analog Input

Analog values can be read through the input channels 1 to 4 (for C-862, 1 to 3) with a resolution of 8 bits.

n indicates the input channel. If *n*=0, all 3 or 4 values are reported.

n = 5 gives the joystick voltage (range 0 to 5 V), and n=6 the button state (not supported for C-862).

Examples:

TA1	Report: **TA1:195**
TA	Report: **TA1:185** **TA2:190** **TA3:220** **TA4:230**

TB Tell Board Address

Reports the “Board Number” = Device Number minus 1 of the currently selected Mercury™ controller.

Device Numbers (1-16) are typically set with the first 4 DIP switches (0-15, negative logic). See the hardware User Manual for more information.

TB	**B:0**

Address selection codes use board numbers, not device numbers.

TC*n* Tell Channel

Reads the digital I/O input on channel *n*. See the hardware user manual for pinout.

(Digital channels can also be read with the single character command “#”)

Parameter *n*:	Digital I/O 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:

TC1	**H01:1** (input high) **H01:0** (input low)
TC0	**H01:F** (all 4 channels high) **H01:5** (channels 1 & 3 high, 2 & 4 low)

TD Tell Dynamic target

DC motor versions only: 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.

Examples:

Command:	Report
TD	**"N:+0000126317"**

TE Tell Error

DC motor versions only: Reports the position error of the motor, as determined by subtracting the actual position from the target position.

Examples:

Command:	Report
TE	**"E:+0000000015"**
TE,WA200,RP99	The report is delivered 5 times a second for 100 times

TF Tell Following error

DC motor versions only: 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 motion parameters.

If the programmed 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-controller 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

TI Tell Iterations

Reports the current value of the repeat counter. It can be used in a repeat loop to determine how many loops are still to perform.

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.

Report1:	**"X:+0000000000"** (First TI is executed before the number of loops is specified!)
Report2	**"X:+0000000099"** and so on.

TL Tell Acceleration

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

Example:

Command:	Report
TL	**L:+0000170000**

See also: SA (Set Acceleration)

TM[*n*] Tell Macros (0 ≤ n ≤ 31)

Displays one or 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.

Example:

Command:	Report
TM1	**MC001 MR55555**
TM0	**MC001 MR55555** **MC002 MA120000**

See also: TZ (Tell Macro 0)

TP Tell Position

Reports the absolute position of the motor. TP may be used to monitor motion during both motor ON and motor OFF status. With stepper motor versions, only commanded motion is accounted for.

(Also implemented as single-character command " ' ", apostrophe)

Command:	Report
TP	**"P:+0000005555"**
TP,WA100,RP	causes the controller to report the position every 100 ms

TS Tell Status

Reports the status of the system, its motion and limit switch states.

(Also implemented as single character command "%")

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

Byte 1	1st character	Bit 4: Macro running Bit 5: Motor OFF Bit 6: Brake ON Bit 7: Drive current active
	2nd character	Bit 0: Ready Bit 1: On target Bit 2: Reference drive active Bit 3: Joystick ON
Byte 2	1st character	Bit 4: Digital input 1 Bit 5: Digital input 2 Bit 6: Digital input 3 Bit 7: Digital input 4
	2nd character	Bit 0: Limit negative Bit 1: Reference signal Bit 2: Limit positive Bit 3: no function
Byte 3	Error numbers	0: No Error 1: RS-232 timeout 2: RS-232 overflow 3: Macro storage full 4: Macro out of range 5: Wrong macro command 6: Command error

Example:

Command:	Report
TS	**"S:04 02 00 03 02 00"**

TT Tell Target

Reports the target position. This is the absolute position to which the motor was commanded. During motion this target may differ from the dynamic target used by the controller to maximize conformance to the proper motion profile (see the TF command, p. 37)

The target position may be specified directly with the MA (Move Absolute) and several other commands, or indirectly with the MR (Move Relative) command.

Example:

Command:	Report
TT	**"T:+0000005000"**

TV Tell current velocity

Reports the current velocity. During motion, outside the acceleration and deceleration phases, the value reported will equal the velocity programmed with SV.

Example:

Command:	Report
TV	**V:+0000016777**

See also: TY, SV

TY Tell programmed velocity

Reports the current programmed velocity setting (not the current velocity). This value can be changed with the SV command.

See also: TV, SV

TZ Tell macro zero

Reads the autostart macro (Macro #0).

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

Example:

Command:	Report
TZ	**MC000 MR5000,WS100,GH** CR+LF+ETX

See also: RZ

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.

Example:

Command:	Report
TL	**"L:+0000900000"**

VE Version report

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

Example:

Command:	Report
VE	**" (c)2007 Physik Instrumente(PI) Karlsruhe, C-863, Ver. 1.09, 2007-08-28"**

WA*n* Wait Absolute (0 < *n* < 65,535)

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

Example:

Command:	Function
MR2000,WA3000,MR-2000	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.
MR2000,WS3000,MR-2000	This command line will move the motor by 2,000 steps, then, after terminating the move it waits for 3000 ms until it moves back for 2000 counts.

WF*n* Wait for channel n OFF

Waits for digital I/O channel *n* to be OFF.

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

Example:

Command:	Function
WF2	This command waits until input channel 2 is OFF (low)

WN*n* Wait for channel n ON (range of n: 1...4)

Waits for digital I/O channel *n* to be ON.

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

Example:

Command:	Function
WN2	This command waits until input channel 2 is ON (high)

WS*n* Wait for motor stop

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 1000 ms is used.

Examples:

Compound commands	Function
MR5000,WS100,RP	Moves 5000 counts/steps, then waits until the motor has reached its target, then waits for another 100 ms before repeating the command line

XF*n* Execute if OFF

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

Example:

Command:	Function
XF3,MR5000	This command moves the motor by 5000 counts/steps only if input channel #3 is low (GND).

See hardware User Manual for number and pinout of digital inputs

XN*n* Execute if ON

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

Example:

Command:	Function
XN3,MR5000	This command moves the motor by 5000 counts/steps only if input channel #3 is high (+5 V).

See hardware User Manual for number and pinout of digital inputs

YF Execute if OFF

Refers to joystick button #1.

Execute the remainder of a command line only if the joystick button connected to joystick button input #1 is OFF (0 V).

See the hardware User Manual for connecting a joystick with 1 or 2 controllers.

Example:

Command:	Function
YF,MR5000	Moves the motor by 5000 counts/steps only if the correctly associated joystick button is not pressed.

YN Execute if ON

Execute the remainder of a command line only if the joystick button connected to joystick button input #1 is ON (+5V).

See the hardware User Manual for connecting a joystick with 1 or 2 controllers.

Example:

Command:	Function
YF,MR5000	Moves the motor by 5000 counts/steps only if the correctly associated joystick button is pressed.

5 Macro Storage on Controller

Macros can be a most powerful tool for the programmer. A *macro command* is a Base Command or a Compound Command stored nside the Controller. Up to 32 macros can be stored in non-volatile memory on each Mercury™ Class controller.

5.1 Features

The native-command macro storage facility has the following features:

- Each macro can contain up to 16 base commands
- Macros are identified by the numbers 0 to 31
- Macro 0, if defined, is the Autostart Macro, which is executed automatically upon power-up or reset
- Macros are executed on the controller where they are stored, so commands in a macro may address only the axis and/or I/O channels associated with that controller (there is no command-interface communication between controllers). Interaction between separate axes is conceivable only through suitable programming and hardwiring of I/O lines
- Once started, a controller macro can continue even if the controller is in the *deselected* state

5.2 Basic Macro Operation

5.2.1 Defining a Macro

To define a macro command, use an MD*n* (Macro Definition) command as the *first* instruction in a compound command followed by a comma-separated list of base commands. *n* sets the macro number.

Note that the system gives no warning if you define (and overwrite) an existing macro.

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.

5.2.2 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.

5.2.3 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 "EMn" where n is the number of the macro.

5.2.4 Limitations

Up to 32 macro sequences, each holding up to 16 base commands, can be stored in the Mercury™ Class 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 section).

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.

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. It is sometimes desirable to define a complex motion sequence in a macro which calls one macro to set important parameters such as torque, gain, or velocity, then calls one or more others to perform the motion.

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

NOTE

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 further macro calls. Up to 32 macro sequences can be stored.

If execution of any of the commands is to be conditional, depending on a run-time condition, then those commands must be grouped together at the end of a macro (see the xxx command descriptions). This restriction makes the use of the EM (execute macro) command in macros even more useful.

5.2.5 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 “power-up default” 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,RP4CR

Macro #0 can be erased by:

RZCR

The contents of macro zero can be read by

TZCR

5.3 Stand-Alone Operation Examples

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-862C-862.PB3), provides virtually unlimited operational flexibility.

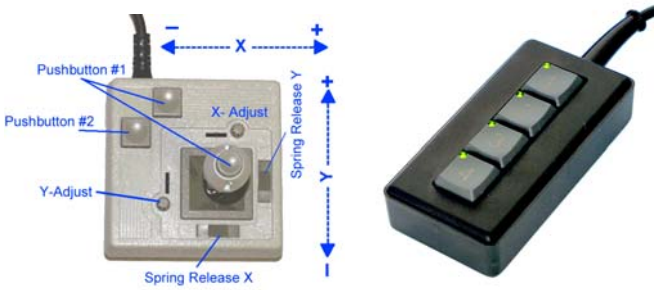


Fig. 1:C-819.30 Joystick and C-170.PB Pushbutton box with LEDs for Mercury™ Controllers

5.3.1 Macros Under Pushbutton/Joystick Control

Mercury™ motor controllers offer convenient manual motion control by using pushbuttons and analog joysticks (joystick not supported by C-862).

!

CAUTION

Do not enable a joystick axis here when no joystick is connected to the controller hardware. Otherwise the corresponding controller axis may start moving and could damage your application setup.

With the joystick mode enabled, both *selected* and *deselected* Mercurys™ can be joystick-operated. The two joystick axes can be connected through the C-819.20Y cable to two Mercury™ controllers.

When a joystick is connected directly to the controller (not to the host PC), it is the *velocity* (not the position or motion of the target) of the controlled axes that is determined by the joystick position.

NOTE

Before a joystick can be operated correctly, a calibration routine may need to be performed. Activating the joystick before calibration may cause the motor to start moving even though the joystick is in the neutral position. To calibrate a joystick axis turn the corresponding “Adjust“ knob on the joystick until the motor stops. If using MMCRun this procedure is facilitated by clicking *Adjust* in the *Joystick* pane. See the *MMCRun* software manual for details.

The C-170.PB pushbutton box connects to the Mercury™ I/O connector. It allows applying TTL signals to the input lines and displays the state of the output lines on LEDs.

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 to10,000 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,EMxx

[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

Mercury™Move 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 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/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.